

ВЫПУСК **2**:

*справочная книга
программиста*

ПЕРСОНАЛЬНЫЕ КОМПЬЮТЕРЫ

ПОД РЕДАКЦИЕЙ Г.Р. ГРОМОВА

СЕРГЕЙ АВАКОВ ОБУЧАЕТ

*детали
настройки*

DOS 3.3-DOS 4

Вирусы и антивирусы

в компьютерах ряда IBM PC

Конфликт **BETA** и **Lansmart**:

как его решить?

"Конопать и промазывать?"

*Элементы
подготовки
винтестера...*



Галина Гнездилова:

МОЯ БОРЬБА

*ЗА НАЦИОНАЛЬНЫЕ
АЛФАВИТЫ
персонального
компьютера*

ПРЕДИСЛОВИЕ РЕДАКТОРА

“БИТ” был задуман редакцией как одно из специализированных изданий библиотеки журнала “Компьютерный мост”. Но, как это часто случается в издательской практике, после выхода в свет первого номера “БИТ” начал жить своей собственной жизнью и вскоре сложился как независимый журнал, зарегистрированный в Госкомпечати СССР 3 декабря 1990 года за номером 1043. Учредителем журнала “БИТ” стал Центр гуманитарной информатики “ИнфоАрт”.

Тематическая направленность первого выпуска “БИТ” была достаточно широка: от этюдов практического программирования в MS-DOS до экономики... Китая периода ранней компьютеризации. Тематика второго “БИТ” резко сужена, и журнал будет в дальнейшем ориентирован на практические потребности широкого круга читателей, ежедневно имеющих дело с персональными ЭВМ: пользователей, программистов, разработчиков.

В значительной степени эта переориентация - прямой отклик авторского коллектива и редакции на письма читателей, среди которых, как это нередко бывает в редакционной почте, встречаются и пожелания, высказываемые в самой энергичной, подчас резкой, форме, как например письмо т. Комова В.В. из Владивостока, фрагменты которого мы приводим ниже, сохранив орфографию и пунктуацию автора:

“Ув. т.т. “Ратекс“

Москву, ул. Дегунинская д. 1 / 4

...

1. ... нечего терять время на исторические рассуждения.

(Анализ 3-х лет PS / 2).

2. Скажите им (авторам и редакции - *прим. ред.*), что ни в библиотеках страны ни в магазинах, (конечно) НЕТУ достаточно полного описания (толковым языком) MS-DOS (Microsoft DOS)

1а. Скажите... , что нужно, и побыстрее (так как уже есть Softwer) толковое пособие для Microsoft Works 2.0 (russ)...

Очень просим: Не тяните...”

К сожалению, мы не имеем возможности даже упомянуть в этом кратком предисловии и небольшую часть наших корреспондентов, чьи пожелания были разносторонне полезны в нашей работе.

Редакция признательна читателям “БИТ” за любые критические замечания и предложения по дальнейшему развитию журнала. Наш адрес: 127486, Москва, ул. Дегунинская, 1 / 4, РАТЕКС / ИнфоАрт.

СПРАВОЧНИК ПРОГРАММИСТА

*По многочисленным просьбам читателей
БИТ начинает публиковать "Руководство
пользователя ПЭВМ", посвященное наиболее
популярным операционным системам:
MS-DOS (версии 3.30 и 4.0) и
OS/2 (версия 1.2)*

УДК 681.3

С.М.Аваков

НАСТРОЙКА ОПЕРАЦИОННОЙ СРЕДЫ В СИСТЕМАХ DOS 3.3 - DOS 4.0

1. ВВЕДЕНИЕ

Настройка операционной среды - первая процедура, с которой сталкивается пользователь, прежде чем он сможет приступить к решению своих прикладных задач. Выполнение подобной процедуры обуславливается многообразием решаемых задач и наличием большого количества различных конфигураций технических средств, с одной стороны, и необходимостью использования единой операционной системы (ОС), с другой. Существуют два подхода к этой проблеме. Первый подход предусматривает поставку ОС в виде дистрибутивного комплекта, содержащего исходные тексты и объектные модули программ, из которых в процессе так называемой процедуры генерации формируется конкретная реализация системы. Второй подход, реализованный в системах DOS*, состоит в том, что настройка системы проводится только путем модификации структур данных, а также манипуляций с готовыми загрузочными модулями программ.

В ОС DOS имеются средства, позволяющие автоматизировать процесс настройки операционной среды. К ним относятся:

- наборы команд для конфигурирования и автоматической настройки системы с помощью файлов CONFIG.SYS и AUTOEXEC.BAT соответственно;
- язык описания пакетных процедур настройки.

* Здесь и далее под DOS будем понимать обобщенное название для системы MS-DOS и PC-DOS.

Ниже приводится описание набора команд, предназначенных для настройки и конфигурирования ОС DOS 3.3 и DOS 4.0.

2. КОНФИГУРИРОВАНИЕ СИСТЕМЫ С ПОМОЩЬЮ ФАЙЛА CONFIG.SYS

Процедура конфигурирования ОС во время перезагрузки, выполняемая под управлением файла с описанием параметров резидента, реализована в системах MS-DOS и PC-DOS начиная с версии 2.0. Файл с описанием параметров резидента называется файлом конфигурирования системы. Спецификация этого файла определена по умолчанию и не может быть изменена пользователем. Файл конфигурирования должен иметь имя CONFIG.SYS и находиться в корневом каталоге магнитного диска, с которого производится перезагрузка.

В процессе конфигурирования устанавливаются параметры и программы, использование которых связано с необходимостью модификации структур данных резидента ОС. Например, для инициализации загружаемого драйвера внешнего устройства (команда DEVICE) требуется переустановка адресов в цепном списке программ обслуживания ввода-вывода (драйверов), для установки количества зарезервированных имен приводов магнитных дисков (команда LASTDRIVE) - модификация базовых данных драйвера магнитных дисков и т. д.

Команды файла CONFIG.SYS анализируются и выполняются программой конфигурирования, расположенной в модуле расширения базовой системы ввода-вывода (BIOS), который находится в скрытом системном файле с именем IO.SYS в системе MS-DOS или IBMBIO.COM в системе PC-DOS.

Программа конфигурирования выполняется автоматически каждый раз при загрузке ядра ОС. Средства запуска этой программы с клавиатуры не предусмотрены, поэтому инициировать процесс конфигурирования можно только путем перезагрузки ОС в целом (путем выполнения функции SYSTEM RESET).

Файл CONFIG.SYS обрабатывается построчно в режиме интерпретации. Если какая-нибудь строка этого файла не воспринимается программой конфигурирования как команда, то на консоль выводится сообщение об ошибке:

```
Unrecognized command in CONFIG.SYS
Error in CONFIG.SYS line N
(Нераспознанная команда в файле CONFIG.SYS
Ошибка в CONFIG.SYS, строка N).
```

В случае, если имя команды конфигурирования задано корректно, но в процессе ввода команды была допущена синтаксическая ошибка при описании параметров, сообщение системы будет иметь следующий вид:

Bad command or parameters - XXX
Error in CONFIG.SYS line N
(Некорректная команда или параметр - XXX
Ошибка в CONFIG.SYS, строка N).

В приведенных сообщениях об ошибках N представляет собой номер строки файла CONFIG.SYS, а XXX - контекст описания параметра, содержащего ошибку. В версии 3.3 сообщение Error in CONFIG.SYS line N не выводится. Это касается всех сообщений об ошибках в командах файла CONFIG.SYS.

По завершении процесса конфигурирования системы образ программы конфигурирования в оперативной памяти затирается и высвобожденная память передается системе для свободного распределения.

По системным соглашениям файл CONFIG.SYS формируется как символьный файл формата ASCII. Такой файл может быть введен или отредактирован с клавиатуры с помощью любого редактора текста или команды COPY. Каждая строка этого файла - это команда конфигурирования, состоящая из двух частей, разделенных знаком равенства (параметры некоторых драйверов также могут содержать знак равенства). До и после знака равенства, отделяющего название команды от параметров, допускается наличие пробелов. Исключением из этого правила является только команда REM, реализованная в DOS 4.0 и предназначенная для включения комментариев в текст файла автонастройки. Синтаксисом этой команды знак равенства не предусмотрен.

Числовые параметры, размещенные в файле CONFIG.SYS, интерпретируются как десятичные числа. Использование параметров с признаком шестнадцатеричного числа "H" не допускается.

Каждая строка файла CONFIG.SYS должна заканчиваться символами <CR> и <LF> (перевод строки и возврат каретки - ASCII 0DH и 0AH). В конце файла обычно помещается символ CTRL-Z (ASCII 1AH), выполняющий в DOS функцию признака конца символьного файла, но наличие этого символа в данном случае не обязательно.

В файл CONFIG.SYS для версий DOS 3.3 могут быть включены следующие команды конфигурирования системы

(точнее, появление этих команд в файле CONFIG.SYS не приведет к выводу сообщения Unrecognized command in CONFIG.SYS): BREAK, BUFFERS, COUNTRY, DEVICE, DRIVPARM, FCBS, FILES, LASTDRIVE, SHELL, STACKS.

В версии 4.0 реализованы три дополнительные команды: SWITCHES, INSTALL, REM.

Замечание. В версиях DOS 2.0 существовала команда SWITCHAR (установка кода символа, выполняющего функции ключа для указания параметров в командной строке). Эта команда была реализована как недокументированная (не описанная в эксплуатационной документации фирм-поставщиков системы), и использовать ее, как и все недокументированные возможности, не рекомендуется. Однако если разрабатываемую программу не предполагается доводить до уровня программного продукта, то использование недокументированных возможностей допустимо. Для пользователей версий 3.X и выше команда SWITCHAR не имеет практической ценности, но описание этой команды предоставляет дополнительную информацию об алгоритме анализа командной строки в системах DOS и о назначении функции 37H прерывания INT 21H, поэтому также приводится в данном разделе.

2.1. Команда BREAK (установка/опрос программного переключателя CTRL-BREAK)

ОБЩИЕ СВЕДЕНИЯ. При работе с компьютером часто возникают ситуации, когда еще до окончания какой-либо программы появляется необходимость ее преждевременного завершения. Сервис, позволяющий выполнять подобные операции, имеется в том или ином виде во всех ОС. В системах DOS (как и во многих других) преждевременное завершение программ инициируется с помощью ввода с клавиатуры кода CTRL-C, получившего, согласно своему назначению, название CTRL-BREAK.

Для обеспечения дополнительной гибкости в системах DOS предусмотрена возможность управления режимом прерывания программ с помощью специального программного переключателя CTRL-BREAK, управлять которым можно как при конфигурировании системы, так и оперативно путем ввода команды с клавиатуры.

НАЗНАЧЕНИЕ. Команда BREAK позволяет разрешить или запретить расширенный режим прерывания процесса выполнения программ с клавиатуры с помощью программного

переключателя CTRL-BREAK. Команда BREAK, заданная без параметров, выводит текущее состояние переключателя.

ОПИСАНИЕ. Прерывание программ по CTRL-BREAK происходит независимо от текущего состояния переключателя, если на клавиши CTRL и BREAK (C) нажимать в момент ожидания терминального ввода или во время вывода на экран консольного терминала (разумеется, если прерываемая программа пользуется системными запросами ввода-вывода).

После установки переключателя BREAK в состояние ON зона действия спецсимвола CTRL-C расширяется. При этом клавиатура опрашивается на CTRL-C при выполнении каждой системной операции ввода-вывода. В частности, прерывание программ возможно в момент выполнения операций обмена данными с устройствами внешней памяти на магнитных дисках. Поэтому при установленном режиме BREAK ON нужно соблюдать осторожность, помня о том, что прерывание программы в момент записи на диск может привести к пропаданию информации, появлению потерянных блоков, ошибок размещения, а также к другим нарушениям файловой структуры магнитных дисков. Такие нарушения могут происходить в некоторых случаях, несмотря на то, что в процессе выполнения функции 00H прерывания INT 21H, инициируемой при исключении программ, система пытается предохранить пользователя от возможных неприятностей:

- аннулирует все операции обмена с магнитными дисками;
- закрывает все файлы, открытые исключаемой программой;
- модифицирует те элементы каталога, которые требуется вернуть в исходное состояние.

Команда BREAK реализована с помощью функции 33H прерывания INT 21H - получение или установка режима исключения программ по CTRL-BREAK. Эта функция позволяет установить нужное значение программного переключателя CTRL-BREAK или получить его состояние. Поэтому программа, выполняющая команду BREAK, очень проста и состоит практически только из двух обращений к этой функции.

Обработка прерывания по CTRL-BREAK выполняется путем прерывания DOS INT 23H.

ФОРМАТЫ.

Ф о р м а т 1
BREAK ON

Ф о р м а т 2
BREAK = ON

Ф о р м а т 3
BREAK OFF

Ф о р м а т 5
BREAK

Ф о р м а т 4
BREAK = OFF

Форматы 1, 3, 5 предназначены для ввода команды с клавиатуры. Появление команд, заданных в этих форматах, в файле CONFIG.SYS вызовет сообщения об ошибках.

Параметр ON разрешает прерывание программ путем одновременного нажатия клавиш <CTRL> <BREAK> или <CTRL> <C> не только во время обмена информацией с консольным терминалом, принтером и портом AUX, но и при выполнении других системных процедур ввода-вывода (за исключением функций 06H и 07H прерывания INT 21H).

Параметр OFF запрещает прерывание программ путем одновременного нажатия клавиш <CTRL> <BREAK> или <CTRL> <C>, если прерываемая программа не находится в состоянии ожидания ввода с клавиатуры или не выводит информацию на экран консольного терминала, принтер или порт AUX. Режим OFF устанавливается по умолчанию после загрузки ОС (разумеется, если в файле CONFIG.SYS не была задана команда BREAK ON).

Если в командной строке специфицирован параметр, отличный от ON или OFF, то выводится сообщение об ошибке:

Must specify ON or OFF

(Нужно специфицировать ON или OFF).

Замечания.

1. Исключить реакцию на ввод кода CTRL-BREAK можно путем программирования ввода с клавиатуры с помощью функций 06H, 07H прерывания INT 21H - транспарентное чтение с клавиатуры консольного терминала. Однако при этом нужно помнить, что обработку всех вспомогательных функций при вводе пользовательская программа должна взять на себя. Аналогичного эффекта можно достичь путем переустановки вектора прерывания INT 23H на программу обработки прерывания, включающую единственную команду возврата из прерывания, или на программу, реализующую какой-нибудь специальный алгоритм обработки данной ситуации.

2. По CTRL-BREAK не может быть завершена программа, запущенная по схеме Terminate & Stay Resident (завершиться и остаться резидентным), реализованной функцией 31H прерывания INT 21H и прерыванием INT 27H. Следует отметить,

что вышеуказанными функциями нельзя инициализировать программы обработки прерываний по CTRL-BREAK, так как при выходе в DOS по этим функциям восстанавливаются векторы прерываний DOS INT 22H - INT 24H (включая вектор 23H, используемый для обработки прерываний по CTRL-BREAK). Разумеется, эти ограничения можно обойти, если воспользоваться нестандартными приемами программирования.

Пример 1.

Для иллюстрации действия программного переключателя CTRL-BREAK рассмотрим прерывание процесса сканирования магнитного диска на наличие вирусов различных типов, который выполняется одной из наиболее распространенных программ этого класса - SCAN McAfee Associates. После запуска этой программы по команде

```
C:\>scan c:>prot.txt
```

все ее сообщения будут протоколироваться в файле PROT.TXT, расположенном на магнитном диске. Сообщения на терминал при этом выводиться не будут. Если перед запуском программы был установлен режим BREAK ON, то ее выполнение можно будет прервать в любой момент при нажатии CTRL-C.

Пример 2а.

```
C:\>break  
C:\>BREAK is off  
C:\>break on  
C:\>break  
C:\>BREAK is on  
C:\>
```

Пример 2б.

BREAK = ON

Замечания к примерам. В примере 2а расширенный режим прерываний по CTRL-BREAK устанавливается заданием команды с клавиатуры консольного терминала. До и после этого вводятся запросы на получение справки о состоянии программного переключателя режима прерываний по CTRL-BREAK.

В примере 26 приводится команда установки расширенного режима прерываний по CTRL-BREAK из файла CONFIG.SYS.

2.2. Команда BUFFERS (установка размера пула буферов оперативной памяти)

ОБЩИЕ СВЕДЕНИЯ. Оптимальный размер пула буферов зависит от многих факторов. Основные из них следующие:

- интенсивность запросов ввода-вывода на выполнение операций обмена с магнитными дисками;
- тип операций доступа к данным (последовательный или прямой);
- объем имеющейся оперативной памяти;
- емкость накопителей на магнитных дисках и физическая структура носителей (количество магнитных поверхностей, дорожек и секторов, информационная емкость сектора);
- характеристики контроллеров дисков (фактор чередования, пропускная способность внутреннего интерфейса, наличие КЭШ-памяти для операций обмена с диском и т.п.);
- пропускная способность системной магистрали;
- быстродействие центрального процессора;
- время цикла оперативной памяти и наличие КЭШ-памяти данных и команд;
- версия и реализация используемой ОС, структура и размер таблицы размещения файлов.

Учитывая многообразие факторов, влияющих на выбор оптимального значения размера пула буферов, а также преимущественно нелинейный характер зависимостей этого значения от каждого из факторов, размер пула буферов оптимизируют, как правило, экспериментальным путем. Опытные пользователи, однако, могут устанавливать это значение по интуиции достаточно близким к оптимальному.

НАЗНАЧЕНИЕ. Команда BUFFERS позволяет установить количество буферов, резервирующихся ОС в оперативной памяти для промежуточной буферизации при выполнении операций обмена информацией с устройствами внешней памяти на магнитных дисках.

ОПИСАНИЕ. Промежуточной буферизацией ввода-вывода сокращаются до минимума издержки, имеющиеся при выполнении операций ввода-вывода в силу различной скорости обработки информации устройствами, участвующими в обмене.

Чаще всего эти издержки выражаются в потере производительности, но в некоторых случаях могут приводить и к потере данных. При обмене данными с магнитными дисками промежуточная буферизация позволяет сократить количество физических операций обмена, заменив часть из них операциями пересылки данных из оперативной памяти, что существенно уменьшает время обмена с магнитными дисками без проведения изменений в аппаратной части.

Часть операций прямого доступа к данным заменяется операциями последовательного доступа. Поэтому уменьшается количество операций позиционирования головок на начало считываемого сектора, а эти операции, как известно, наиболее длительны по времени. Для накопителей на жестких дисках, например, среднее время позиционирования головок на произвольный цилиндр составляет ориентировочно 20...60 мс, среднее время поиска сектора на дорожке (время половины оборота диска) - порядка 10 мс, а время формирования накопителем шестнадцатиразрядного слова данных измеряется микросекундами. При пятнадцатисекторной разметке, например, ориентировочное время считывания одной дорожки в зависимости от размера промежуточного буфера оперативной памяти может изменяться примерно в два раза. Буферизация ввода-вывода будет наиболее эффективной в том случае, если объем буфера оперативной памяти синхронизирован с размером физической дорожки или цилиндра магнитного диска.

Уменьшение размера буфера по сравнению с размером дорожки эквивалентно увеличению фактора чередования, обеспечиваемого контроллером магнитных дисков. Поэтому имея, например, дорогой контроллер дисков, обеспечивающий возможность работы с фактором чередования, равным единице, и неправильно установив размер буфера обмена, можно свести к нулю эффективность дополнительных затрат на покупку высокопроизводительной аппаратуры.

Следует отметить, что чрезмерное увеличение размера буфера приводит к увеличению времени поиска нужной информации в оперативной памяти. На компьютерах типа XT минимальной конфигурации, например, замедление работы наблюдается уже начиная примерно с пятидесяти буферов. С другой стороны, если размер буфера увеличен по сравнению с размером дорожки, но не кратен ему (либо, наоборот, размер дорожки не кратен размеру буфера), то это неизбежно приведет к снижению эффективности использования буфера.

ФОРМАТ.

BUFFERS = X

Параметр X - число зарезервированных буферов. Задаваемое значение X может лежать в пределах 1...255. Если эта команда не содержится в файле CONFIG.SYS, то система автоматически определяет число буферов в зависимости от объема имеющейся в наличии оперативной памяти и емкости накопителей на магнитных дисках:

минимальная аппаратная конфигурация.....	2
накопитель емкостью более 360 Кбайт	3
объем оперативной памяти 128...255 Кбайт	5
объем оперативной памяти 256...511 Кбайт	10
объем оперативной памяти 512 Кбайт и более	15
конфигурация PC XT жестким диском 10 Мбайт	16
конфигурация PC AT с жестким диском 32 Мбайт и более	32

Замечания.

1. Не рекомендуется устанавливать размер буфера меньше размера таблицы размещения файлов FAT (File Allocation Table), так как эта таблица считывается при каждом обращении к диску и ее разделение на части в оперативной памяти существенно увеличивает количество физических обращений к магнитным дискам и время доступа к файлам.

2. В версии 4.0 в формат команды BUFFERS дополнительно введены параметр Y и ключ /X:

BUFFERS = X[,Y][/X]

Параметром Y устанавливается максимальный размер буфера, используемого в отдельной операции ввода-вывода; по умолчанию - 1.

Ключ /X позволяет использовать для буферизации расширенную память. При заданном ключе /X параметр X может принимать значения до 10000.

П р и м е р 3.

BUFFERS = 32

2.3. Команда COUNTRY (установка характеристик, специфичных для страны)

ОБЩИЕ СВЕДЕНИЯ. В процессе адаптации ОС к условиям работы различных групп пользователей обычно возникает много мелких проблем, связанных с национальным языком и некоторыми условными обозначениями, традиционными для дан-

ной страны (например, обозначение знака денежной единицы). Команда COUNTRY в совокупности с набором команд, предназначенных для управления таблицами кодов символов, позволяет автоматически, без модификации программного и аппаратного обеспечения, решать этот комплекс проблем для систем DOS.

Следует отметить, что решение данного вопроса было бы невозможно без наличия определенной гибкости в аппаратных средствах, в частности без перезагружаемых знакогенераторов терминальных устройств.

НАЗНАЧЕНИЕ. Командой COUNTRY можно задавать характеристики системы, специфичные и традиционные для каждой страны. Такими характеристиками являются:

- формат даты;
 - конфигурация символа денежной единицы;
 - разделитель триад числа (разделитель тысяч);
 - разделитель десятичного числа, отделяющий целую часть числа от его дробной части (разделитель десятичного числа и разделитель триад числа всегда представлены парой символов, один из которых - точка, а другой - запятая);
 - символы, используемые в качестве разделителей элементов даты и времени;
 - формат задания текущего часа (диапазон изменения переменной, обозначающей текущий час, 1...12 или 0...23);
 - количество десятичных значащих разрядов при описании чисел с символом денежной единицы;
 - формат денежной единицы (расположение знака денежной единицы относительно числа, обозначающего денежную сумму);
- разделитель списков данных.

ОПИСАНИЕ. Характеристики, специфичные для страны, помещаются в специальный блок оперативной памяти, который называется блоком информации страны. Структура этого блока имеет следующий вид:

00Н	Последовательность указания элементов даты: 0 - месяц, день, год; 1 - день, месяц, год; 2 - год, месяц, день.
02Н	Код символа, используемого в качестве знака денежной единицы (задается в формате ASCIIZ).
07Н	Код символа-разделителя триад десятичного числа (ASCIIZ).
09Н	Код символа, разделяющего десятичное число на целую и дробную части (ASCIIZ).

0BH	Код символа, разделяющего элементы даты (ASCIIZ).
0DH	Код символа, разделяющего элементы времени (ASCIIZ).
0FH	Формат денежной единицы: 0 - \$XXX.XX; 1 - XXX.XX\$; 2 - \$XXX.XX; 3 - XXX.XX \$; 4 - XXX\$XX.
10H	Количество значащих десятичных разрядов при описании чисел со знаком денежной единицы.
11H	Формат задания текущего часа: 0 - [1...12]; 1 - [0...23].
12H	Адрес вызова Case Map, используемого для переназначения кодов ASCII, превышающих 80H.
16H	Код символа, используемого в качестве разделителя списков данных (ASCIIZ).
18H	Резервное поле.
20H	Размер блока информации страны в байтах.

Получить данные из этого блока или записать данные в него из пользовательской программы можно с помощью функции 38H прерывания INT 21H.

Командой COUNTRY устанавливается весь блок вышеприведенных параметров по заданному коду страны в процессе конфигурирования системы под управлением файла CONFIG.SYS. При интерпретации команды COUNTRY система выбирает информацию, соответствующую указанной стране, из файла с именем COUNTRY.SYS. В принципе, пользователь может подготовить свой файл со специфической информацией и использовать механизм, предоставляемый командой COUNTRY и функцией 38H прерывания INT 21H, для своих целей. Однако при этом нужно помнить, что ОС в процессе работы будет использовать ту же самую информацию, что может привести к изменению текста и алгоритма синтаксического контроля некоторых входных сообщений (таких, как время, дата и т. п.).

ФОРМАТ.

COUNTRY = XXX[,YYY][,*спец. файла*]

Параметр XXX - трехзначный код выбранной страны. По умолчанию принимается код 001 (USA). Если специфицированный код не поддерживается используемой версией ОС, то на консоль выводится сообщение:

Invalid country code
Error in CONFIG.SYS line N
(Некорректный код страны
Ошибка в CONFIG.SYS, строка N).

Параметр YYY - трехзначный код таблицы кодов символов. По умолчанию принимается 437. В случае ошибки на консоль выводится сообщение:

Invalid country code or code page
Error in CONFIG.SYS line N
(Некорректный код страны или таблица кодов
Ошибка в CONFIG.SYS, строка N).

Параметр *спец.файла* - спецификация файла, содержащего информацию о стране. Эта спецификация может включать имя устройства, маршрут, имя и расширение имени файла. По умолчанию используется файл с именем COUNTRY.SYS из корневого каталога на текущем приводе магнитных дисков. Если спецификация файла с информацией о стране задана неверно, то на консольный терминал выводится сообщение об ошибке:

Bad or missing "имя файла"
Error in CONFIG.SYS line N
(Некорректно задан файл "имя файла"
Ошибка в CONFIG.SYS, строка N).

В случае нарушения структуры параметров команды COUNTRY на консольный терминал выводится следующее сообщение об ошибке:

Error in country command
Error in CONFIG.SYS line N
(Ошибка в команде COUNTRY
Ошибка в CONFIG.SYS, строка N).

П р и м е р 4.

COUNTRY = 049

2.4. Команда DEVICE (включение в систему загружаемого драйвера)

ОБЩИЕ СВЕДЕНИЯ. Система управления вводом-выводом DOS - это гибкая децентрализованная система, позволяющая унифицировать алгоритмы обмена как со стандартными, так и с нестандартными внешними устройствами компьютера. Она состоит из общей программы, осуществляющей диспетчирова-

ние процессов ввода-вывода, и набора отдельных программ-драйверов, каждая из которых управляет конкретным внешним устройством.

Существуют два типа драйверов - резидентные и загружаемые. Резидентные драйверы находятся в составе ядра ОС и являются его неотъемлемой частью. К ним относятся драйверы, обслуживающие базовую конфигурацию компьютера с минимальным набором выполняемых функций, такие как драйвер дисков, системной консоли, таймера и т.п.

Загружаемые драйверы, как правило, выполняют функции, необходимые не всем пользователям. Каждый пользователь может подобрать набор драйверов, оптимальный для его задач и конфигурации компьютера. Такой подход существенно сокращает издержки оперативной памяти на управление вводом-выводом и соответственно размер резидента ОС.

Кроме того, пользователь может разрабатывать загружаемые драйверы нестандартных внешних устройств (или стандартных устройств, работающих по нестандартным алгоритмам). Если при разработке собственных драйверов пользователем будет выполнен ряд системных соглашений, то система возьмет на себя часть функций по диспетчированию ввода-вывода для устройства пользователя и организации его программного интерфейса.

НАЗНАЧЕНИЕ. По команде **DEVICE** могут устанавливаться как стандартные загружаемые драйверы внешних устройств, поставляемые с системой, так и нестандартные драйверы, разработанные пользователем или приобретенные у других пользователей-поставщиков программного обеспечения. С системами **DOS 3.3** поставляются следующие загружаемые драйверы:

ANSI.SYS	- расширенный драйвер клавиатуры;
DISPLAY.SYS	- драйвер, поддерживающий режим переключения таблиц кодов консольного терминала;
DRIVER.SYS	- драйвер внешнего привода гибких дисков произвольного формата;
PRINTER.SYS	- драйвер, поддерживающий режим переключения таблиц кодов для устройств печати;
RAMDRIVE.SYS	- драйвер, поддерживающий электронный диск (в ОС PC-DOS этот драйвер имеет имя VDISK.SYS).

С версией 4.0 дополнительно поставляется драйвер SMARTDRV.SYS, поддерживающий режим кэширования данных при обмене с жесткими дисками в основной, расширенной или отображаемой оперативной памяти.

Поддержка отображаемой памяти. В состав DOS во многих случаях включаются также драйверы, обеспечивающие возможность работы с расширенной и отображаемой оперативной памятью в рамках стандарта LIM EMS (Lotus, Intel, Microsoft Expanded Memory Specification - стандарт фирм Лотус, Интел, Майкрософт для работы с отображаемой памятью). Ниже будут рассмотрены два таких драйвера:

- | | |
|-------------|--|
| XMA2EMS.SYS | - программа управления отображаемой памятью (Expanded Memory Manager), обеспечивающая поддержку стандарта LIM EMS; |
| XMAEM.SYS | - этот драйвер эмулирует работу адаптера отображаемой памяти процессора 80286 Expanded Memory Adapter/A в системах с процессором 80386 и обеспечивает возможность работы драйвера отображаемой памяти XMA2EMS.SYS. |

Примерами аппаратных средств, обеспечивающих работу в режиме LIM EMS с помощью драйвера XMA2EMS.SYS, могут служить устройства, совместимые со следующими адаптерами:

- IBM 80286 Expanded Memory Adapter/A;
- IBM 80286 Memory Expansion Option;
- IBM 2 MB Expanded Memory Adapter;
- IBM PS/2 mod.80 (или другие компьютеры на базе 80386) с установленным драйвером XMAEM.SYS.

Следует отметить, что версия 4.0 стандарта LIM EMS не только вобрала в себя лучшие качества разработанных ранее стандартов LIM EMS 3.2 и EEMS (стандарт для работы с отображаемой памятью фирм AST Research, Quadram и Ashton Tate), но и имеет ряд преимуществ перед ними, благодаря чему получила широкое распространение. Основные отличия LIM EMS 4.0 от LIM EMS 3.2:

- существенно расширен набор функций (30 вместо 15);
- возможность работы со значительно большими объемами оперативной памяти (32 Мбайт вместо 8 Мбайт);

- возможность выполнения программ из отображаемой памяти.

Во многих случаях драйверы, предназначенные для программной поддержки стандарта LIM EMS, поставляются фирмами-изготовителями вместе с платами памяти. В названии этих драйверов, как правило, используется аббревиатура EMM (Expanded Memory Manager - программа управления отображаемой памятью). В качестве примеров таких аппаратных средств можно привести следующие устройства:

- Above Board Plus фирмы INTEL;
- Vocaram/AT Plus фирмы Boca Research;
- AST Rampage Plus 286 фирмы AST Research;
- EMS Memory Board фирмы Micron Technology.

ОПИСАНИЕ. Выполнение команды DEVICE начинается с загрузки драйвера в оперативную память и анализа типа устройства в пятнадцатом разряде слова атрибутов устройства, расположенного в третьем слове заголовка драйвера. Единичное значение этого разряда соответствует внешним устройствам, ориентированным на блоки, а нулевое - устройствам, ориентированным на символы. К первому типу относятся устройства, обеспечивающие прямой доступ к информации и обменивающиеся блоками данных, размеры которых диктуются физическими принципами работы этих устройств (например, магнитные диски). Ко второму типу - устройства, с которыми возможен только последовательный обмен информацией с передачей за одну операцию ввода-вывода одного байта или одного слова данных.

В оперативной памяти драйверы располагаются в следующей последовательности:

- 1) драйвер NUL-устройства;
- 2) загружаемые драйверы устройств, ориентированных на символы;
- 3) драйвер системной консоли CON;
- 4) драйвер порта AUX;
- 5) драйвер печати PRN;
- 6) драйвер таймера CLOCK;
- 7) драйвер магнитных дисков;
- 8) другие резидентные драйверы внешних устройств, ориентированных на символы или блоки;
- 9) загружаемые драйверы устройств, ориентированных на блоки.

Драйверы внешних устройств, ориентированных на блоки, загружаются в конец списка после ранее загруженных драйверов устройств, ориентированных на блоки; драйверы

устройств, ориентированных на символы, - после драйвера NUL-устройства в начало списка, перед ранее загруженными драйверами внешних устройств, ориентированных на символы. Поэтому загружаемые драйверы символьных устройств имеют более высокий приоритет обслуживания запросов ввода-вывода и могут шунтировать существующие в системе стандартные резидентные и загружаемые драйверы.

В процессе инициализации загружаемого драйвера устанавливается соответствующее поле связи драйвера (четырехбайтный указатель на следующий драйвер) и таким образом драйвер оказывается включенным в цепной список адресов драйверов внешних устройств. Этот список используется в дальнейшем системой при обработке запросов ввода-вывода для нахождения устройства, которому предназначается поступивший запрос. Для последнего драйвера в поле связи записываются два слова, содержащее каждое из которых равно "1". Эти два слова выполняют функцию признака конца списка. В заключение производится функция инициализации драйвера (функции с кодом 0), которая возвращает адрес первой свободной ячейки оперативной памяти, расположенной после тела драйвера. На этом процесс загрузки драйвера заканчивается.

ОБОБЩЕННЫЙ ФОРМАТ

DEVICE = *спец.файла* [ключи]

Параметр *спец.файла* - спецификация файла, содержащего загрузочный модуль подключаемого драйвера. Спецификация включает имя привода, маршрут, имя и расширение имени файла. Имя привода и маршрут могут задаваться по умолчанию. Умолчания для имени и расширения имени файла не предусмотрено. Если имя файла драйвера не задано или задано некорректно, то система выдает следующее сообщение об ошибке:

Bad or missing filename

Error in CONFIG.SYS line N

(Отсутствует или некорректно задан файл "имя файла"

Ошибка в CONFIG.SYS, строка N)

Поле *ключи* содержит параметры, предназначенные для настройки подключаемого драйвера. Это поле может содержать несколько ключей и параметров и, как правило, является уникальным для каждого драйвера. Ниже приведены форматы

команды DEVICE для каждого из загружаемых драйверов, используемые при инициализации этих драйверов под управлением файла CONFIG.SYS.

ANSI.SYS - РАСШИРЕННЫЙ ДРАЙВЕР КЛАВИАТУРЫ

Драйвер ANSI.SYS обеспечивает программную поддержку расширенного набора команд управления экраном и клавиатурой консольного терминала (CON), задаваемых с помощью ESC-последовательностей. Драйвер ANSI.SYS включается в состав систем PC-DOS и MS-DOS начиная с версии 2.0.

ФОРМАТ.

DEVICE = [d:][*маршрут*]ansi.sys[/X]

Ключ /X устанавливает режим, в котором драйвер позволяет осуществлять раздельное переназначение функций одинаковых клавиш, расположенных на различных полях клавиатуры (например, клавиша END в цифровом поле и в поле управления курсором на 101-клавишной или 102-клавишной клавиатуре). Ключ /X реализован только в версии 4.0.

DISPLAY.SYS - ДРАЙВЕР ТАБЛИЦ КОДОВ СИСТЕМНОЙ КОНСОЛИ

Драйвер DISPLAY.SYS обеспечивает программную поддержку режима переключения таблиц кодов консольного терминала (CON). Он входит в состав систем PC-DOS и MS-DOS начиная с версии 3.3.

ФОРМАТ.

DEVICE = [d:][*маршрут*]display.sys con[:] =
(*тип* [, *антк*][, N, M])

Параметр *тип* задает тип адаптера дисплея. Допустимые значения - MONO, CGA, EGA и LCD. Если в командной строке параметр *тип* опущен, то драйвер сам тестирует адаптер дисплея и определяет его тип. В версии 4.0 при указании параметра EGA можно работать с двумя типами адаптеров - EGA и VGA.

Параметр *антк* (аппаратно-поддерживаемая таблица кодов) задает таблицу кодов, поддерживаемую аппаратно. Допустимы следующие значения:

437 - США;
850 - многоязыковая таблица;
860 - Португалия;
863 - Канада (французская);
865 - Норвегия.
866 - Россия

Параметр N задает число поддерживаемых дополнительных таблиц кодов (0...12) в зависимости от аппаратных средств. Для MONO и CGA с не реализованной поддержкой дополнительных фонтов это количество должно быть равно 0. Для адаптеров EGA или LCD значение N может быть равно 2 или 1 соответственно.

Параметр M задает число субфонтов, поддерживаемых для каждой таблицы кодов.

В случае ошибки в команде инициализации драйвера DISPLAY.SYS система выдает сообщение о ней:

Invalid syntax on DISPLAY.SYS
Error in CONFIG.SYS line N
(Некорректный синтаксис для DISPLAY.SYS
Ошибка в CONFIG.SYS, строка N).

DRIVER.SYS - ДРАЙВЕР ВНЕШНИХ ПРИВОДОВ МАГНИТНЫХ ДИСКОВ

Драйвер DRIVER.SYS управляет внешними приводами устройств внешней памяти.

ФОРМАТ.

DEVICE = [d:][*маршрут*]driver.sys/d:*номер*[/C][/F:*фактор*][/N][/H:*головки*][/S:*секторы*][/T:*дорожки*]

Параметр /d:*номер* задает номер физического привода в пределах 0...255 (для версии 4.0 - 0...127). При этом номеру 0 соответствует первый привод гибких дисков (привод A), номеру один - второй привод (привод B), номеру два и далее - внешние приводы.

Ключ /C указывает на необходимость контроля состояния замка привода гибких магнитных дисков, а также наличия дискеты в кармане накопителя.

Ключ /F:*фактор* специфицирует тип привода магнитных дисков (фактор форматирования). Параметр *фактор* может задавать следующие типы приводов:

- 0 - гибких дисков 5,25" емкостью 160/180 Кбайт или 320/360 Кбайт;
- 1 - гибких дисков 5,25" емкостью 1,2 Мбайт;
- 2 - гибких дисков 3,5" емкостью 720 Кбайт;
- 3 - гибких дисков 8" одинарной плотности;
- 4 - гибких дисков 8" двойной плотности;
- 5 - жесткого диска;
- 6 - магнитной ленты;
- 7 - гибких дисков 3,5" емкостью 1,44 Мбайт.

По умолчанию устанавливается значение 2.

Ключ /H:*головки* задает максимальное число головок, которое может быть установлено в диапазоне 1...99. По умолчанию принимается 2.

Ключ /S:*секторы* задает число секторов на дорожке 1...99. По умолчанию принимается 9.

Ключ /T:*дорожки* задает число дорожек на магнитной поверхности 1...999. По умолчанию принимается 80.

Ключ /N специфицирует неперегружаемое устройство, обменивающееся блоками, например фиксированный (жесткий) диск.

Замечания.

1. В версии 4.0 ключ /N и значения 3, 4, 5 и 6 ключа /F упразднены.

2. Драйвер DRIVER.SYS входит в состав систем MS-DOS и PC-DOS начиная с версии 3.2.

PRINTER.SYS - ДРАЙВЕР ТАБЛИЦ КОДОВ ПРИНТЕРОВ

Драйвер PRINTER.SYS обеспечивает программную поддержку режима переключения таблиц кодов печатающего устройства. Драйвер PRINTER.SYS включен в состав систем PC-DOS и MS-DOS начиная с версии 3.3.

ФОРМАТ.

DEVICE = [d:][*маршрут*]printer.sys lptx = (mtn[,anmx[...]][,N])

Параметр *mtn* задает тип используемого принтера. Допустимыми значениями этого параметра (разумеется, что допу-

скается использование не только оригинальных, но и совместимых принтеров):

- 4201 - IBM 4201 Proprinter;
IBM 4202 Proprinter XL;
- 4208 - IBM 4207 или 4208;
IBM Proprinter X24 или XL24;
- 5202 - IBM 5202 Quietwriter III.

Параметр *аптк* (аппаратно-поддерживаемая таблица кодов) задает требуемую таблицу кодов, поддерживаемую аппаратно. Допустимы следующие значения:

- 437 - США;
- 850 - многоязыковая таблица;
- 860 - Португалия;
- 863 - Канада (французская);
- 865 - Норвегия.

Параметр *N* задает число добавочных таблиц кодов, поддержка которых может осуществляться.

В случае ошибки в команде инициализации драйвера PRINTER.SYS система выдает сообщение о ней:

Invalid syntax on PRINTER.SYS
(Некорректный синтаксис для PRINTER.SYS).

RAMDRIVE.SYS - ДРАЙВЕР ЭЛЕКТРОННОГО ДИСКА

Драйвер RAMDRIVE.SYS (VDISK.SYS) предназначен для обеспечения доступа к оперативной памяти в режиме электронного диска. Драйвер RAMDRIVE.SYS (VDISK.SYS) включен в состав системы MS-DOS (PC-DOS) начиная с версии 3.2.

ФОРМАТ.

DEVICE = ramdrive.sys [*рдск*][*рсект*][*ркорн*][*/E*][*/A*]

Параметр *рдск* задает размер эмулируемого диска в килобайтах. По умолчанию принимается 64 Кбайт. Минимальный размер составляет 16 Кбайт.

Параметр *рсект* специфицирует размер сектора в байтах. По умолчанию устанавливается 128 байт (в системе 4.0

по умолчанию принимается 512). Разрешены следующие значения: 128, 256, 512 и 1024 байт.

Параметр *ркорн* - это количество записей, зарезервированных в корневом каталоге эмулируемого диска. По умолчанию устанавливается 64. Может задаваться в диапазоне 4...1024. При инициализации драйвер увеличивает специфицированное количество зарезервированных записей, расширяя корневой каталог до границы сектора.

Ключ /С позволяет использовать отображаемую память (свыше 1 Мбайт). Ключ /С нельзя использовать совместно с ключом /А.

Ключ /А позволяет использовать отображаемую память с помощью стандарта LIM EMS. Ключ /А нельзя использовать совместно с ключом /Е.

SMARTDRV.SYS - ДРАЙВЕР КЭШИРОВАНИЯ ДИСКОВ

Драйвер SMARTDRV.SYS управляет режимом кэширования жестких дисков в расширенной или внешней оперативной памяти. Драйвер SMARTDRV.SYS включен в состав системы MS-DOS (PC-DOS) только для версии 4.0.

ФОРМАТ.

DEVICE = [d:][*маршрут*]smartdrv.sys [*размер*][/А]

Параметр *размер* задает размер памяти, используемой драйвером. По умолчанию устанавливается 256 Кбайт.

Ключ /А позволяет использовать отображаемую оперативную память или внешнюю оперативную память.

XMA2EMS.SYS - ДРАЙВЕР LIM EMS

Драйвер XMA2EMS.SYS управляет отображаемой памятью (Expanded Memory Manager).

ФОРМАТ.

DEVICE = [d:*маршрут*]xma2ems.sys FRAME = *адрес*
P254 = *адрес1* P255 = *адрес2* /S:*размер*

Параметр FRAME = *адрес* устанавливает адрес области памяти, содержащей такое число страниц оперативной памяти размером по 16 Кбайт, которое указано параметром *размер*

при ключе /S. Если ключ /S не указан, то устанавливается число страниц, равное 4.

Ключ /S:*размер* позволяет определить размер используемой области памяти в страницах по 16 Кбайт.

Параметр P254 = *адрес1* указывает адрес специальной страницы памяти, резервируемой для драйвера RAMDRIVE.SYS (VDISK.SYS) и программы FASTOPEN.

Параметр P255 = *адрес2* указывает адрес специальной страницы памяти, резервируемой для команды BUFFERS с ключом /X.

ХМАЕМ.SYS - ДРАЙВЕР ЭМУЛЯЦИИ ОТОБРАЖАЕМОЙ ПАМЯТИ

Драйвер ХМАЕМ.SYS эмулирует работу адаптера отображаемой памяти процессора 80286 Expanded Memory Adapter/A. С помощью этого драйвера Вы можете эмулировать отображаемую память (expanded memory), необходимую для реализации стандарта LIM EMS 4.0, на базе расширенной памяти (extended memory).

ФОРМАТ.

DEVICE = [d:*маршрут*]хмаем.sys *размер*

Параметр *размер* задает размер отображаемой памяти, отведенной под эмуляцию отображаемой памяти, в страницах по 16 Кбайт.

П р и м е р 5.

DEVICE = ansi.sys

П р и м е р 6.

DEVICE = driver.sys/d:l/f:l/h:2/s:l5/t:80

П р и м е р 7.

DEVICE = smartdrv.sys 1024/A

П р и м е р 8.

DEVICE = C:\ХМА2EMS.SYS FRAME = D000 P254 = C000
P255 = C400 /X:96

П р и м е р 9.

DEVICE C:\DOS40\XMAEM.SYS

Замечания к примерам. В примерах 5 - 9 показана установка пяти различных загружаемых драйверов:

в примере 5 - драйвера расширенной клавиатуры, позволяющего использовать ESC-последовательности для управления клавиатурой и экраном;

в примере 6 - драйвера для управления внешним приводом гибких магнитных дисков 5,25" емкостью 1,2 Мбайт;

в примере 7 - драйвера, осуществляющего поддержку режима кэширования данных при обмене с магнитными дисками;

в примере 8 - драйвера для работы с отображаемой памятью (expanded memory) объемом 1,5 Мбайт. В этом примере показана также установка специальных страниц оперативной памяти для работы драйвера RAMDRIVE.SYS (VDISK.SYS) и команды FASTOPEN, используемой с ключом /X (P254), а также для работы команды BUFFERS, заданной с ключом /X (P255);

в примере 9 - драйвера расширенной памяти (extended memory), позволяющего использовать эту память для эмуляции отображаемой памяти (expanded memory).

2.5. Команда DRIVPARM (установка характеристик привода магнитных дисков)

ОБЩИЕ СВЕДЕНИЯ. По команде DRIVPARM определяются параметры блочного внешнего устройства, программная поддержка которого будет осуществляться драйвером DRIVER.SYS.

НАЗНАЧЕНИЕ. Команда DRIVPARM устанавливает характеристики внешних устройств, обменивающихся блоками, таких как приводы магнитных дисков, магнитных лент и т.д.

ОПИСАНИЕ. По команде DRIVPARM драйвер DRIVER.SYS настраивается на конкретное блочное устройство путем модификации соответствующих внутренних структур данных драйвера.

ФОРМАТ.

DRIVPARM /d:номер = [/C][/F:тип_устр][/H:кол_пов]
[/N][/S:кол_сект][/T:кол_дор]

Спецификация /d:номер задает номер привода, для которого устанавливаются характеристики. При этом приводу А: соответствует номер 0, приводу В: - 1 и т.д.

Ключ /С указывает на необходимость контроля состояния замка привода гибких дисков и наличия дискеты в кармане накопителя.

Ключ /F:тип_устр определяет тип формата устройства внешней памяти на магнитных дисках:

- 0 - гибкий диск 5" (133 мм) емкостью 160/180 Кбайт или 320/360 Кбайт;
- 1 - гибкий диск 5" (133 мм) емкостью 1,2 Мбайт;
- 2 - гибкий диск 3,5" (89 мм) емкостью 720 Кбайт;
- 3 - гибкий диск 8" (203 мм) с одинарной плотностью записи;
- 4 - гибкий диск 8" (203 мм) с двойной плотностью записи;
- 5 - жесткий диск;
- 6 - накопитель на магнитной ленте;
- 7 - гибкий диск 3,5" (89 мм) емкостью 1,44 Мбайт.

По умолчанию устанавливается тип 2, что соответствует приводу гибких магнитных дисков емкостью 720 Кбайт размером 3,5 дюйма (89 мм).

Ключ /H:Кол_пов позволяет задать максимальное число магнитных поверхностей (количество магнитных головок) внешнего устройства (привода магнитных дисков). Это число может принимать значения от 1 до 99. По умолчанию устанавливается значение 2.

Ключ /N служит признаком фиксированного устройства (например, жесткого диска).

Ключ /S:кол_сект задает число секторов на дорожке (от 1 до 99). По умолчанию устанавливается значение 9.

Ключ /T:кол_дор задает число дорожек на каждой магнитной поверхности (от 1 до 999). По умолчанию устанавливается значение 80.

Замечания.

1. В версии 4.0 значения 3 и 4 ключа /F упразднены. Дополнительно введен ключ /I, позволяющий подключать привод 3,5" гибких дисков без поддержки ROM-BIOS (в ком-

пьютерах, совместимых с PC-XT, а также в некоторых компьютерах, совместимых с PC-AT, такая поддержка отсутствует).

2. Ошибка при задании команды **DRIVPARM** может привести к нарушению или к полному прекращению работы ОС.

3. Перед выполнением команды описания произвольного блочного устройства или перед инициализацией драйвера **DRIVER.SYS** следует убедиться в наличии свободного имени устройства и при необходимости выполнить команду **LASTDRIVE**.

4. Пользователям, не имеющим глубоких знаний о системе, применять команду **DRIVPARM** не рекомендуется.

Пример 10.

DRIVPARM = /D:4/F:6/H:1/S:99/T:100

В примере 10 устанавливаются параметры привода магнитной ленты. После конфигурирования системы в качестве имени устройства будет использоваться имя **E**.

2.6. Команда **FCBS** (установка числа открытых блоков **FCB**)

ОБЩИЕ СВЕДЕНИЯ. Команда **FCBS** устанавливает характеристики процесса диспетчирования файловой системы. Подобная необходимость чаще всего возникает при использовании команды установки режима разделенного доступа к файлам **SHARE**.

НАЗНАЧЕНИЕ. По команде **FCBS** определяется число блоков управления файлами **FCB**, которые могут быть открыты одновременно в режиме разделенного доступа к файлам, а также задается число блоков **FCB**, которые ОС не может закрыть автоматически.

ОПИСАНИЕ. Когда в системе достигается максимально возможное количество одновременно открытых файлов (установленное параметром **X** команды **FCBS**), при поступлении новых запросов на открытие файлов система автоматически закрывает файлы, которые дольше всего находятся в открытом состоянии. В некоторых случаях это может привести к существенному увеличению системных издержек. В частности, такая ситуация снизит эффективность кэширования данных при обмене с дисками. Для предотвращения автоматического за-

крытия файлов в системе предусмотрена возможность блокирования произвольного числа файлов в открытом состоянии. Однако при использовании этого параметра нужно помнить, что немотивированное увеличение количества заблокированных файлов может привести к нежелательным последствиям, в частности к увеличению времени ожидания запросов на обмен данными с устройствами внешней памяти.

Правильный выбор параметров команды FCBS увеличивает эффективность системы при параллельной обработке запросов.

ФОРМАТ.

FCBS = X,Y

Параметр X задает максимальное количество одновременно открытых файлов в пределах 1...255. По умолчанию X устанавливается равным четырем.

Параметр Y задает число файлов, заблокированных от автоматического закрытия при достижении лимита одновременно открытых файлов, заданного параметром X. Значение параметра Y - в пределах 1...255. Оно не может быть больше значения параметра X. По умолчанию это значение устанавливается равным 0.

Пр и м е р 11.

FCBS = 8,2

2.7. Команда FILES (установка количества логических номеров устройств)

ОБЩИЕ СВЕДЕНИЯ. Начиная со второй версии DOS, в системе реализован универсальный механизм доступа к файлам и внешним устройствам - система доступа по логическим номерам устройств/файлов (File handles). Необходимое количество логических номеров при этом определяется степенью разветвленности процессов ввода-вывода. В свою очередь от количества используемых логических номеров устройств существенно зависят издержки оперативной памяти на организацию такого механизма доступа. На каждый дополнительный логический номер внешнего устройства/файла в резиденте системы требуется около 50 байт оперативной памяти.

НАЗНАЧЕНИЕ. Команда FILES позволяет сделать издержки оперативной памяти минимальными. Эта команда устанавливает максимально возможное число одновременно открытых файлов, доступ к которым осуществляется по логическим номерам внешних устройств/файлов.

ОПИСАНИЕ. Параметр X команды FILES задает размерность системной таблицы открытых файлов (system open-file table). Максимальная размерность этой таблицы - 255 (00FFH), определяет верхнее ограничение параметра X. Для каждого отдельного процесса по умолчанию предусмотрена возможность работы с 20 открытыми файлами одновременно (если это число не превышает установку, сделанную по команде FILES).

При загрузке любой программы пять номеров оказываются занятыми под стандартные устройства. В дальнейшем при открытии каждого нового файла число номеров увеличивается на единицу. Этот процесс продолжается до тех пор, пока общее число открытых файлов не достигнет 20 или не заполнится системная таблица открытых файлов.

Начиная с версии DOS 3.3 в системе имеются средства, позволяющие устранить ограничение номеров. Доступ к этим средствам осуществляется с помощью функции 67H прерывания INT 21H - установить счетчик файлов, которая отслеживает таблицу логических номеров устройств, сформированную в PSP (program segment prefix - префикс программного сегмента) пользовательской программы. Требуемое число логических номеров устройств задается в регистре BX и соответственно разрядности этого регистра может устанавливаться до 65534. Однако максимальное число одновременно открытых файлов ограничено размером системной таблицы открытых файлов и поэтому не может быть больше 255. Максимальное число логических номеров внешних устройств/файлов (File handles), используемых одной программой, составляет 65534.

ФОРМАТ.

FILES = X

Параметр X задает число логических номеров устройств, используемых одновременно, в пределах 8...255. По умолчанию X устанавливается равным 8.

Логические номера от 0 до 4 применяются для управления стандартными внешними устройствами системы:

Логический номер устройства	Физическое имя устройства по умолчанию	Логическое имя устройства	Тип внешнего устройства
0	CON:	stdin	клавиатура
1	CON:	stdout	экран
2	CON:	stderr	экран
3	AUX:	stdaux	устройство aux
4	PRN:	stdprn	принтер

Поэтому, чтобы получить количество логических номеров, имеющих в распоряжении пользователя, от общего числа нужно отнять пять номеров.

Таким образом, при установке параметра X по умолчанию свободными остаются только три логических номера, что обычно бывает недостаточно. Чаще всего в системах среднего класса в файле CONFIG.SYS резервируется 20 логических номеров устройств /файлов:

FILES = 20

Если в процессе конфигурирования системы при интерпретации файла CONFIG.SYS значение параметра X больше 255 или меньше 8, на экран консольного терминала выдается сообщение об ошибке:

Bad command or parameters - X
Error in CONFIG.SYS line N
(Некорректная команда или параметры - X
Ошибка в CONFIG.SYS, строка N).

Если установленного числа логических номеров устройств в процессе инициализации операционной среды оказывается недостаточно, то на экран консольного терминала выводится сообщение

EXEC failure
(Ошибка при выполнении).

Появление этого сообщения означает, что в команде FILES необходимо увеличить параметр X.

П р и м е р 12.

FILES = 16

2.8. Команда LASTDRIVE (установка количества зарезервированных имен приводов магнитных дисков)

ОБЩИЕ СВЕДЕНИЯ. Командой LASTDRIVE устраняется избыточность в структурах данных, используемых при обмене с магнитными дисками. Однако если проанализировать, высвобождение каких объемов оперативной памяти может быть достигнуто, то не трудно видеть, что речь идет о долях процента от общего объема. Поэтому во многих случаях при конфигурировании системы устанавливают максимально возможное количество зарезервированных имен приводов магнитных дисков.

Следует отметить, что в некоторых случаях экономия даже такого небольшого объема памяти может оказаться существенной с точки зрения высвобождения виртуального адресного окна в диапазоне шестнадцатиразрядного адреса и будет составлять ориентировочно 1...20% имеющегося свободного пространства оперативной памяти, которое остается после загрузки резидента ОС.

Учитывая вышеуказанное обстоятельство, а также просто руководствуясь соображениями здравого смысла, количество зарезервированных имен обычно устанавливают на несколько единиц большим количества приводов магнитных дисков, имеющих в конфигурации компьютера. Небольшой запас имен резервируется на случай возникновения необходимости создания логических приводов на жестком диске, виртуального диска в оперативной памяти, замены маршрутов с помощью команды SUBST и т. д.

НАЗНАЧЕНИЕ. Командой LASTDRIVE устанавливается необходимое количество зарезервированных в системе имен приводов магнитных дисков в пределах 1...26 - по числу букв английского алфавита.

ОПИСАНИЕ. В процессе выполнения команды LASTDRIVE в оперативной памяти формируются структуры данных, необходимые для программной поддержки заданного количества накопителей на магнитных дисках. Получить количество зарезервированных в системе имен приводов можно с помощью функции 0H прерывания INT 21H. Количество зарезервированных имен, возвращаемых этой функцией в AL, совпадает с количеством, установленным командой LASTDRIVE. Значением по умолчанию является 5 (соответствует значению параметра *имя привода* команды LASTDRIVE, устанавливаемому по умолчанию).

ФОРМАТ.

LASTDRIVE = *имя привода*

Параметр *имя привода* специфицирует последнюю из букв латинского алфавита, используемую для обозначения имени привода магнитных дисков (с этой целью могут быть использованы все буквы английского алфавита - от буквы А до буквы, указанной в данной команде). В качестве умолчания используется Е, максимально возможное значение - Z.

Замечания.

1. Команда LASTDRIVE не воспринимается при попытке зарезервировать в системе количество имен, меньшее, чем количество имеющихся приводов магнитных дисков. Система резервирует столько имен, сколько в наличии логических и физических приводов. Сообщение об ошибке при этом не выдается.

2. При увеличении количества зарезервированных имен возрастает (хотя и несущественно) размер резидента ОС, поэтому, как уже упоминалось, не имеет смысла резервировать количество имен приводов, существенно превышающее количество используемых физических и логических устройств. Следует помнить, что для каждого резервируемого имени привода система использует 80 байт оперативной памяти на расширение системных таблиц. При резервировании более 16 имен накопителей на магнитных дисках (значение параметра *имя привода* - Q) к суммарному объему издержек добавляется еще 16 байт.

3. Затраты оперативной памяти на резервирование имен устройств не меняются при значениях параметра *имя привода* от А до F.

П р и м е р 13.

LASTDRIVE = A
LASTDRIVE = B

LASTDRIVE = Z

В примере 13 приведены все возможные варианты команды LASTDRIVE на момент, когда в системе существовало шесть приводов магнитных дисков.

При изменении параметра *имя привода* от А до F объем издержек оперативной памяти не меняется, так как в этих случаях значение параметра LASTDRIVE устанавливается по умолчанию.

2.9. Команда SHELL (установка интерпретатора командной строки)

ОБЩИЕ СВЕДЕНИЯ. Команда SHELL меняет характеристики интерфейсной части ОС, касающейся связи с оператором. С помощью команды SHELL можно загрузить либо нестандартный интерпретатор команд, написанный пользователем (командный процессор верхнего уровня), либо стандартный интерпретатор COMMAND.COM с измененными параметрами (размером окружения, маршрутом по умолчанию и т. д.).

В первом случае появляется возможность выбора произвольного интерфейса с оператором (интерфейс с оператором будет определяться новым интерпретатором). В качестве примера можно привести программу COMMAND-PLUS ESP Systems.

Второй случай дает некоторую гибкость при установке стандартного интерпретатора (позволяет разместить интерпретатор в новом каталоге, изменить имя файла, размер окружения и т. д.).

НАЗНАЧЕНИЕ. Команда SHELL устанавливает интерпретатор командной строки (командный процессор верхнего уровня) при конфигурировании ОС под управлением файла CONFIG.SYS в процессе загрузки, т.е. задает произвольную спецификацию файла интерпретатора - спецификацию, отличную от стандартной, используемой загрузчиком ОС по умолчанию.

ОПИСАНИЕ. Укрупненный алгоритм работы нестандартного интерпретатора командной строки можно представить следующим образом:

1. Ограничение объема памяти, занимаемой интерпретатором, путем выполнения функции 4AH прерывания INT 21H (модификация блока памяти).

2. Установка переменной COMSPEC, представляющей спецификацию файла исполняемой копии командного процессора в окружении DOS.

3. Переустановка вектора прерываний по вводу кода преждевременного завершения программы CTRL-BREAK.

4. Вывод подсказки интерпретатора на стандартное устройство вывода.

5. Ввод команды пользователя со стандартного устройства ввода.

6. Выделение из содержимого буфера командной строки символьной константы от начала буфера до первого пробела.

7. Поиск в таблице имен внутренних команд команды с совпадающим именем.

8. Поиск в текущем и альтернативных каталогах внешней команды с заданным именем.

9. Вывод сообщения об ошибке или выполнение команды.

10. Циклическое повторение шагов 4 - 9.

После выполнения команды SHELL при загрузке командного процессора система вместо COMMAND.COM будет загружать интерпретатор командной строки, указанный в спецификации спец.файла команды SHELL.

При выполнении команды SHELL устанавливается специальная переменная окружения:

COMSPEC = спец.файла

Значение этой переменной - символьная константа, задающая текущее умолчание для спецификации файла командного процессора.

Переменная COMSPEC просматривается или изменяется с помощью команды SET. Система использует эту переменную при перезагрузке транзитной части командного процессора, поэтому, переустановив переменную COMSPEC по команде SET, можно инициировать загрузку другого командного процессора без перезагрузки ОС (например, по команде COMMAND).

ФОРМАТ.

SHELL = спец.файла[/E:xxxxx][/P]

Параметр *спец.файла* задает спецификацию файла загрузочного модуля интерпретатора команд, включающую имя привода, маршрут, имя и расширение имени файла.

Ключ /E:xxxxx определяет размер окружения ОС в байтах.

Ключ /P делает загружаемый интерпретатор команд резидентным и вызывает после загрузки выполнение пакетного файла автонастройки AUTOEXEC.BAT.

Замечания.

1. Новый интерпретатор команд можно написать таким образом, что вместе с ключами /E и /P (или вместо них) в команде SHELL будут использоваться другие ключи. Поэтому в более общем случае формат командной строки для SHELL представляется так:

SHELL = спец.файла[параметры]

2. По умолчанию в качестве командного процессора система использует файл COMMAND.COM, расположенный в корневом каталоге на приводе, с которого загружается ОС.

П р и м е р 14.

SHELL = \DOS4\COMMAND.COM/E:512/P

П р и м е р 15.

SHELL = \UNIDOS\PLAKS.COM

Замечания к примерам. В примере 14 стандартный процессор COMMAND.COM загружается с расширенной системой области связи. После загрузки выполняется файл AUTOEXEC.BAT и новая версия командного процессора становится резидентной.

В примере 15 показана установка нового командного процессора пользователя, реализующего расширенный набор команд по отношению к стандартному набору команд интерпретатора COMMAND.COM.

2.10. Команда STACKS (установка размера стека)

ОБЩИЕ СВЕДЕНИЯ. Правильное планирование конфигурации аппаратного стека позволяет повысить надежность и эффективность работы системы в целом. Размер стека должен выбираться таким образом, чтобы при пиковых ситуациях глубина стека оказывалась достаточной для размещения необходимой информации. С другой стороны, чрезмерно большой стек ведет к непроизводительным затратам оперативной памяти.

При расчете размера стека учитываются суммарные издержки системных и прикладных программ. Основные пара-

метры, учитываемые при планировании конфигурации стека, следующие:

- глубина вложенности прерываний;
- глубина вложенности подпрограмм;
- издержки на сохранение регистров;
- издержки при обмене параметрами между программами через стек;
- наличие необходимости временного хранения данных в стеке;
- количество параллельных процессов в системе.

Следует отметить, что, хотя в качестве расчетных используются пиковые значения параметров, размер стека рекомендуется устанавливать с запасом в 20...50%.

Если в процессе работы периодически выводится сообщение системы о переполнении аппаратного стека:

Internal Stack Failure
System Halted
(Ошибка внутреннего стека
Система остановлена),

то это чаще всего означает, что глубина стека установлена недостаточной.

НАЗНАЧЕНИЕ. Команда STACKS предназначена для описания конфигурации дополнительных структур аппаратного стека.

ОПИСАНИЕ. Дополнительная возможность при работе со стеком - автоматическое переключение стековых сегментов. Эта возможность весьма полезна при использовании защищенного режима процессора 80286 или при работе с 32-разрядными процессорами 80386 и 80486. За счет гибкого управления стеком повышается эффективность его использования и снижаются издержки на переключение процессов.

Кроме того, благодаря рассматриваемому механизму управления стеком программы обработки прерываний, использующие стек, становятся повторно-входимыми (реентерабельными) и упрощаются процедуры обработки вложенных прерываний. При каждом прерывании аппаратных средств система переустанавливает стековый сегмент, выбирая очередной свободный буфер из зарезервированного по команде STACKS пула. По окончании программы обработки прерывания буфер объявляется свободным и возвращается обратно в пул для последующего перераспределения. Если в качестве аргументов

команды STACKS были установлены нули, то DOS не переключает стек на время обработки прерывания.

ФОРМАТ.

STACKS = N,S

Параметр N задает количество сегментов (используемых в качестве стека), определяемое количеством параллельно выполняемых процессов. Оно может принимать значения 0...64 (для PC-AT не рекомендуется значение меньше 8). По умолчанию для компьютеров класса IBM PC, IBM PC-XT, IBM PC-Portable устанавливается значение 0, а для компьютеров класса IBM PC-AT и выше - значение 9.

Параметр S задает размер каждого стекового сегмента в байтах в интервале 0...512 (для PC-AT не рекомендуется значение меньше 32). По умолчанию для компьютеров класса IBM PC, IBM PC-XT, IBM PC-Portable устанавливается значение 0, а для компьютеров класса IBM PC-AT и выше - значение 128.

Если параметры команды STACKS заданы некорректно, на дисплей выводится сообщение об ошибке:

Invalid STACK parameter
(Некорректный параметр стека).

Замечания.

1. Если параметры N и S установлены равными нулю, то система не переключает стек на время обработки прерывания (используется только один стек).

2. Необходимость в использовании команды STACKS возникает в случаях, если одновременно выполняется несколько программ, обрабатывающих прерывания от внешних устройств.

3. Команду STACKS не рекомендуется применять пользователям, не имеющим глубоких знаний об ОС и архитектуре процессора, так как неправильное планирование размера стека может привести к сложным сбойным ситуациям и нарушению работоспособности системы.

4. Команда STACKS описывает дополнительную часть стека, поэтому нулевые значения параметров команды также допустимы.

Пример 16.

STACKS = 4,256

метры, учитываемые при планировании конфигурации стека, следующие:

- глубина вложенности прерываний;
- глубина вложенности подпрограмм;
- издержки на сохранение регистров;
- издержки при обмене параметрами между программами через стек;
- наличие необходимости временного хранения данных в стеке;
- количество параллельных процессов в системе.

Следует отметить, что, хотя в качестве расчетных используются пиковые значения параметров, размер стека рекомендуется устанавливать с запасом в 20...50%.

Если в процессе работы периодически выводится сообщение системы о переполнении аппаратного стека:

Internal Stack Failure
System Halted
(Ошибка внутреннего стека
Система остановлена),

то это чаще всего означает, что глубина стека установлена недостаточной.

НАЗНАЧЕНИЕ. Команда STACKS предназначена для описания конфигурации дополнительных структур аппаратного стека.

ОПИСАНИЕ. Дополнительная возможность при работе со стеком - автоматическое переключение стековых сегментов. Эта возможность весьма полезна при использовании защищенного режима процессора 80286 или при работе с 32-разрядными процессорами 80386 и 80486. За счет гибкого управления стеком повышается эффективность его использования и снижаются издержки на переключение процессов.

Кроме того, благодаря рассматриваемому механизму управления стеком программы обработки прерываний, использующие стек, становятся повторно-входимыми (реентерабельными) и упрощаются процедуры обработки вложенных прерываний. При каждом прерывании аппаратных средств система переустанавливает стековый сегмент, выбирая очередной свободный буфер из зарезервированного по команде STACKS пула. По окончании программы обработки прерывания буфер объявляется свободным и возвращается обратно в пул для последующего перераспределения. Если в качестве аргументов

команды STACKS были установлены нули, то DOS не переключает стек на время обработки прерывания.

ФОРМАТ.

STACKS = N,S

Параметр N задает количество сегментов (используемых в качестве стека), определяемое количеством параллельно выполняемых процессов. Оно может принимать значения 0...64 (для PC-AT не рекомендуется значение меньше 8). По умолчанию для компьютеров класса IBM PC, IBM PC-XT, IBM PC-Portable устанавливается значение 0, а для компьютеров класса IBM PC-AT и выше - значение 9.

Параметр S задает размер каждого стекового сегмента в байтах в интервале 0...512 (для PC-AT не рекомендуется значение меньше 32). По умолчанию для компьютеров класса IBM PC, IBM PC-XT, IBM PC-Portable устанавливается значение 0, а для компьютеров класса IBM PC-AT и выше - значение 128.

Если параметры команды STACKS заданы некорректно, на дисплей выводится сообщение об ошибке:

Invalid STACK parameter
(Некорректный параметр стека).

Замечания.

1. Если параметры N и S установлены равными нулю, то система не переключает стек на время обработки прерывания (используется только один стек).

2. Необходимость в использовании команды STACKS возникает в случаях, если одновременно выполняется несколько программ, обрабатывающих прерывания от внешних устройств.

3. Команду STACKS не рекомендуется применять пользователям, не имеющим глубоких знаний об ОС и архитектуре процессора, так как неправильное планирование размера стека может привести к сложным сбойным ситуациям и нарушению работоспособности системы.

4. Команда STACKS описывает дополнительную часть стека, поэтому нулевые значения параметров команды также допустимы.

Пример 16.

STACKS = 4,256

В версиях до DOS 3.3 включительно инициализировать такие программы можно было только путем задания соответствующих команд с клавиатуры либо из пакетного файла. В версии 4.0 появилась возможность с помощью команды INSTALL выполнять инициализацию программ этого типа при конфигурировании системы из файла CONFIG.SYS.

НАЗНАЧЕНИЕ. Командой INSTALL устанавливаются программы в процессе конфигурирования системы при загрузке.

ОПИСАНИЕ. Программы типа TSR формируются двумя способами:

- с помощью прерывания INT 27H (размер устанавливаемой программы ограничен 64 Кбайт);
- с помощью функции 31H прерывания INT 21H (предоставляется более общий механизм).

ФОРМАТ.

INSTALL = [d:*маршрут*] *имя файла* [*параметры*]

Спецификация d:*маршрут* задает привод и маршрут для файла загрузочного модуля устанавливаемой программы. Спецификация задается, если загрузочный модуль не находится в корневом каталоге.

Параметр *имя файла* специфицирует имя и расширение имени файла с загрузочным модулем устанавливаемой программы.

Параметры - параметры, с которыми будет установлена программа. Конкретный набор параметров определяется синтаксисом программы.

Замечания.

1. При использовании команды INSTALL указывайте полную спецификацию файла (включая полный маршрут и тип файла).

2. В некоторых ОС под установкой программы понимается процедура получения системой физического номера начального сектора загрузочного модуля, а также некоторых параметров, позволяющих в дальнейшем максимально быстро произвести загрузку и запуск этой программы. Такой процедурой, с одной стороны, экономится оперативная память, а с другой, - достигается наиболее быстрая реакция на запросы об активизации программ. Этот параметр является особенно существенным в системах реального времени.

Пример 18.

INSTALL = C:\D4\FASTOPEN.EXE C:

2.13. Команда SWITCHES (блокировка расширенной клавиатуры)

ОБЩИЕ СВЕДЕНИЯ. Команда SWITCHES применяется для получения совместимости с программным обеспечением, которое не поддерживает расширенную клавиатуру.

НАЗНАЧЕНИЕ. Команда SWITCHES ограничивает применение расширенной клавиатуры базовым набором функций.

ОПИСАНИЕ. Команда SWITCHES шунтирует обработку запросов расширенной клавиатуры.

ФОРМАТ.

SWITCHES = /K

Пример 19.

SWITCHES = /K

2.14. Команда REM (определение строки комментариев)

ОБЩИЕ СВЕДЕНИЯ. Желание наиболее полно использовать возможности ОС, а также необходимость ее расширения часто приводят к появлению достаточно сложных файлов конфигурирования. В таких случаях возникает потребность введения в текст файла конфигурирования комментариев, описывающих некоторые нюансы процедуры конфигурирования. Возможность введения комментариев в файл конфигурирования системы появилась впервые в версии 4.0. Она реализована с помощью команды REM (REMARK - замечание, комментарий, ремарка).

НАЗНАЧЕНИЕ. Команда REM предназначена для включения комментариев в файл конфигурирования системы.

ОПИСАНИЕ. REM - неисполняемая команда. Программа конфигурирования пропускает строки файла CONFIG.SYS, начинающиеся с имени REM.

ФОРМАТ.

REM комментарий

Параметр *комментарий* - произвольная строка текста.

Замечание. REM - это единственная команда конфигурирования системы, синтаксис которой не требует наличия знака равенства между именем команды и параметрами.

Пример 20.

REM - поддержка LIM EMS

3. НАСТРОЙКА СИСТЕМЫ С ПОМОЩЬЮ ФАЙЛА АВТОНАСТРОЙКИ AUTOEXEC.BAT

Средства для создания пакетных файлов команд ОС (косвенных командных файлов) впервые появились в системе PC-DOS версии 1.25. Наличие подобных средств пакетной обработки позволяет существенно расширить спектр сервисных функций ОС, делает ее более мобильной. Основное назначение пакетных файлов - обеспечение возможности настройки операционной среды под конкретные задачи пользователя, что привело, в свою очередь, к реализации специализированного языка высокого уровня, ориентированного на программирование системных процедур.

Операционная среда может настраиваться как автоматически - при загрузке системы, так и вручную - путем запуска пользовательских пакетных файлов с клавиатуры. По системным соглашениям для автоматической настройки загрузчик ОС использует файл с именем AUTOEXEC.BAT, расположенный в корневом каталоге магнитного диска, с которого производится перезагрузка. Файл AUTOEXEC.BAT в отличие от файла CONFIG.SYS может быть запущен с клавиатуры. Для пакетных файлов, запускаемых с клавиатуры, могут использоваться произвольные имена файлов, но в качестве расширения имени пакетного файла должно применяться расширение BAT (от английского batch - пачка, пакет).

Пакетный файл - символьный файл, сформированный в формате ASCII. Такой файл вводится или редактируется с клавиатуры дисплея с помощью любого редактора текста общего назначения или команды COPY.

Каждая строка пакетного файла должна заканчиваться символами <LF> и <CR> (перевод строки и возврат каретки - ASCII 0AH и 0DH). В конце файла обычно помещается символ CTRL-Z (код ASCII 1AH), выполняющий в DOS функцию признака конца символьного файла, но наличие этого символа в пакетном файле не является обязательным условием для работы командного процессора.

Пакетные файлы обрабатываются построчно в режиме интерпретации. Каждая строка пакетного файла представляет собой команду DOS. В пакетный файл могут быть включены любые внутренние или внешние команды DOS (разумеется, за исключением команд, предназначенных для конфигурирования системы). Синтаксический анализ и выполнение команд пакетного файла производятся командным процессором (интерпретатором командной строки).

Для более эффективного использования пакетных файлов в системе предусмотрено несколько служебных команд. Эти команды превращают, по существу, язык описания пакетных файлов в специализированный язык программирования высокого уровня. К числу таких команд относится ECHO, описанная ниже, а также следующие семь команд:

- обращения к подпрограмме из пакетного файла CALL;
- организации цикла в пакетном файле FOR;
- безусловного перехода в пакетном файле GOTO;
- условного перехода в пакетном файле IF;
- приостановки выполнения пакетного файла PAUSE;
- включения комментариев в пакетный файл REM;
- сдвига переменных пакетного файла SHIFT.

Кроме того, при программировании пакетных файлов используются некоторые другие возможности. Ниже дано описание команд DOS, предназначенных для настройки операционной среды.

3.1. Команда CHCP (переключение таблиц кодов символов)

ОБЩИЕ СВЕДЕНИЯ. Команда CHCP (сокращенно CHange Code Page - смена таблиц кодов), впервые реализованная в версии 3.3, - одно из звеньев системы поддержки национальных алфавитов. Общая укрупненная схема системных

средств, используемых для обеспечения такой поддержки, приведена при описании команды NLSFUNC.

Команда CHCP принадлежит системному уровню, т. е. ее действие распространяется на все внешние устройства, поддерживаемые системой. Для них режим переключения таблиц кодов вполне допустим. Забегая вперед, отметим, что функции команды MODE, связанные с переключением таблиц кодов, реализуют уровень внешнего устройства (действие команды распространяется только на внешнее устройство, специфицированное в командной строке).

Команда CHCP полезна, если работа ведется на двух различных языках и требуется оперативное переключение таблиц кодов символов.

НАЗНАЧЕНИЕ. Команда CHCP предназначена для замены текущей таблицы кодов командного процессора COMMAND.COM, используемой для всех внешних устройств системы с реализованной поддержкой загружаемых таблиц кодов, на таблицу кодов, специфицированную в командной строке.

Команда CHCP, заданная без параметров, выводит на экран номер активной таблицы кодов, а также номера привязанных таблиц кодов.

ОПИСАНИЕ. Команда CHCP устанавливает одну из двух привязанных системных таблиц кодов в качестве текущей таблицы кодов. Если специфицированная в команде CHCP таблица кодов не привязана к системе, то выдается следующее сообщение:

```
Code page XXX not prepered for system
Active code page: YYY
Prepared system code pages: YYY ...
(Таблица кодов XXX не привязана к системе)
Активная таблица кодов: YYY
Привязанные к системе таблицы кодов: YYY ...)
```

Если устройство (экран, клавиатура, принтер) не привязано к таблице кодов, система выводит следующее сообщение об ошибке:

```
Code page XXX not prepared for device xxx
(Таблица кодов XXX не привязана к устройству xxx).
```

Для устранения указанных ошибок необходимо установить соответствующие связи между системой, устройством и таблицей кодов. При этом следует убедиться, что программа NLSFUNC резидентна, а для загружаемого драйвера устройства, поддерживающего перезагружаемые таблицы кодов, выполнена привязка таблицы кодов с помощью команды **MODE CODEPAGE PREPARE** и **MODE CODEPAGE SELECT**.

ФОРМАТ.

CHCP [XXX]

Параметр XXX задает номер таблицы кодов, которая станет текущей после выполнения команды. Таблица кодов, специфицированная в командной строке, должна быть привязана к системе и соответствовать команде **COUNTRY** из файла **CONFIG.SYS**, выполненной при загрузке системы.

Номер таблицы кодов XXX принимает следующие значения:

- 437 - США (установлен по умолчанию);
- 850 - многоязыковая таблица;
- 860 - Португалия;
- 863 - Французская Канада;
- 865 - Дания, Норвегия.

Если специфицированная таблица кодов не воспринимается системой, то на дисплей выводится сообщение об ошибке:

Invalid code page specified

(Специфицирована некорректная таблица кодов).

Замечания.

1. При выполнении команды **CHCP** должны быть доступны соответствующие драйверы внешних устройств. Если драйвер устройства не установлен в системе, то функции **CHCP** для этого устройства выполняться не будут.

2. Команда **CHCP** может быть задана только после выполнения команды **NLSFUNC** (программа **NLSFUNC.EXE** должна быть установлена).

3. При выполнении команды **CHCP** может потребоваться доступ к файлу **COUNTRY.SYS**, содержащему информацию, специфичную для страны.

4. Команда **CHCP** внутренняя.

Пример 21.

C:\>chcp 850

C:\>

Пример 22.

C:\>chcp

Active code page: 437

Prepared system code pages: 437, 850

C:\>

Замечания к примерам. В примере 21 в качестве текущей таблицы кодов устанавливается таблица кодов 850, позволяющая работать, например, с учетом особенностей немецкого алфавита.

В примере 22 выводится справочная информация об активной (437) и привязанных к системе (437, 850) таблицах кодов.

3.2. Команда СТТУ (переназначение консоли)

ОБЩИЕ СВЕДЕНИЯ. Команда СТТУ (аббревиатура Change Teletype - смена телетайпа) используется редко. Необходимость в применении этой команды может появиться в случае, если имеется какая-либо специальная конфигурация системы, например по последовательному интерфейсу подключена рабочая станция, и в качестве консоли (системного терминала) используется телетайп.

Следует отметить, что команда СТТУ применяется также для преднамеренного отключения системы от консольного терминала, например при организации защиты системы от несанкционированного доступа (см. примеры команды PROMPT).

Действие команды СТТУ не распространяется на команды, подаваемые из пакетных файлов, что позволяет реализовывать временное отключение системного терминала, например для исключения вывода каких-либо системных сообщений (см. примеры команды СТТУ).

НАЗНАЧЕНИЕ. Команда СТТУ используется для переназначения внешнего устройства, выполняющего функции системного терминала (консольного терминала).

ОПИСАНИЕ. По команде СТТУ переназначаются логические устройства, реализованные на физическом устройстве CON, новому физическому устройству, специфицированному в командной строке команды СТТУ. Такими логическими устройствами являются:

stdin - стандартное устройство ввода (по умолчанию - клавиатура дисплея);

stout - стандартное устройство вывода (по умолчанию - экран дисплея);

sterr - стандартное устройство для вывода сообщений об ошибках (по умолчанию - экран дисплея).

После выполнения команды СТТУ продолжение диалога пользователя с системой возможно только в случае, если в качестве нового консольного устройства (системного терминала) в команде было задано имя дуплексного устройства (устройства ввода-вывода). Исключение составляет только псевдоустройство NUL (нуль-устройство).

В ОС DOS зарезервированы следующие стандартные имена дуплексных устройств:

- | | | |
|------------|---|---|
| AUX (COM1) | - | первый последовательный порт ввода-вывода; |
| COM2 | - | второй последовательный порт ввода-вывода; |
| COM3 | - | третий последовательный порт ввода-вывода; |
| NUL | - | псевдоустройство ввода-вывода, выполняющее функции нуль-устройства, которое является источником (приемником) фиктивных запросов ввода-вывода; |
| CON | - | консольный терминал. |

Имена внешних устройств PRN (LPT1), LPT2, LPT3 зарезервированы под устройства вывода.

ФОРМАТ.

СТТУ *устройство*[:]

Параметр *устройство* задает физическое имя любого внешнего устройства, определенного в системе, которое выполняет функции ручного ввода и визуального вывода символьной информации. Если специфицированное имя не воспринимается, СТТУ выдает сообщение об ошибке:

Invalid device
(Некорректное устройство).

Знак : (двоеточие) - необязательный признак имени внешнего устройства.

Замечания.

1. При использовании программ, в которых ввод-вывод производится на физическом уровне, переназначения команды СТТУ не действуют.

2. Если в качестве консоли указать устройство вывода, например LPT1(принтер), то система “зависнет”. Компьютер в этом случае будет пытаться получить команду от этого порта, что, естественно, невозможно.

3. В случае преднамеренного переназначения консоли на ноль-устройство (NUL) или на устройство вывода нужно помнить, что после выполнения соответствующей команды СТТУ восстановить исходное назначение можно только перезагрузкой ОС или путем подачи команды

СТТУ CON

из уже выполняющегося пакетного файла (если команда СТТУ была задана из пакетного файла).

4. Команда СТТУ внутренняя.

Пример 23.

C:\>ctty aux

C:\>

Пример 24.

СТТУ NUL
XCOPY *.* D:
СТТУ CON

Замечания к примерам. После выполнения команды, приведенной в примере 23, все запросы ввода-вывода будут переназначены и система начнет воспринимать команды с устройства AUX.

В примере 24 приведен текст пакетного файла, позволяющего скопировать группу файлов без вывода сообщений на экран консольного терминала (за счет переназначения консоли на ноль-устройство во время работы команды XCOPY).

3.3. Команда ЕСНО (установка/опрос программного переключателя режима эхо-печати)

ОБЩИЕ СВЕДЕНИЯ. Управление режимом эхо-печати пакетных файлов позволяет, с одной стороны, иметь удобную возможность наблюдения за ходом выполнения пакетных файлов (трассирования пакетных файлов) при их отладке, а с другой, - не загромождать экран лишней информацией и устранить издержки по быстродействию, возникающие за счет дополнительного вывода на экран, при выполнении пакетных файлов.

НАЗНАЧЕНИЕ. Команда ЕСНО выполняет две функции, каждая из которых имеет самостоятельное значение:

- функции управления состоянием программного переключателя эхо-печати команд при выполнении пакетных файлов;
- функции оператора вывода сообщений на экран консольного терминала из пакетных файлов.

Задание команды без параметров позволяет определить текущий режим ЕСНО, установленный в системе (ON - включено или OFF - отключено).

ОПИСАНИЕ. Команды ЕСНО, вызываемые из пакетного файла, по умолчанию протоколируются на экране. Подавить вывод на экран можно с помощью команды ЕСНО OFF.

По окончании выполнения пакетного файла программный переключатель режима ЕСНО устанавливается в исходное состояние (ЕСНО ON).

Следует отметить, что задание команды ЕСНО OFF с клавиатуры ведет к подавлению вывода системной подсказки, что в некоторых случаях также может оказаться полезным (например, когда необходимо ввести длинную командную строку, которая не помещается в одной строке с системной подсказкой).

Если в качестве параметра команды ECHO задана символьная строка, отличная от ON или OFF и не состоящая из одних пробелов, то команда выполняется по алгоритму оператора вывода сообщений. Если в качестве параметра команды заданы значения ON или OFF, команда соответственно включает или выключает режим эхо-сопровождения. Если параметр отсутствует или задана строка, состоящая из пробелов, то команда ECHO выводит на экран информацию о текущем режиме эхо-печати (ON или OFF).

ФОРМАТЫ.

Ф о р м а т 1.

Установка режима эхо-печати на экране консоли команд пакетных файлов.

ECHO ON

Ф о р м а т 2.

Выключение режима эхо-печати на экране консоли команд пакетных файлов.

ECHO OFF

Ф о р м а т 3.

Вывод сообщения на экран консоли.

ECHO *цеп.симв.*

Ф о р м а т 4.

Опрос программного переключателя ECHO.

ECHO

Параметр ON вызывает включение режима отображения выполняемых команд пакетного файла на экране дисплея. Этот режим устанавливается по умолчанию.

Параметр OFF выключает режим отображения выполняемых команд пакетного файла на экране дисплея.

Параметр *цеп.симв.* задает цепочку символов, представляющую собой сообщение, которое будет выводиться на экран дисплея независимо от текущего режима ECHO. Таким образом, формат 3 позволяет использовать команду ECHO в качестве оператора вывода символьной информации на экран консольного терминала из пакетных файлов.

Замечания.

1. С помощью команды ЕСНО с использованием символов переадресации ввода-вывода можно выводить символьные последовательности на произвольное устройство вывода. Например, команда:

```
echo Файл для справочной информации> spr.txt
```

повлечет создание в текущем каталоге нового файла с именем SPR.TXT и запись в него строки текста: “Файл для справочной информации”.

2. При отключении режима эхо-печати команд пакетных файлов текст самой команды ЕСНО:

ЕСНО OFF

выводится на экран консольного терминала. Для подавления этого вывода перед командой нужно ввести префикс “@”. Текст команды в этом случае должен выглядеть следующим образом:

@ЕСНО OFF

Префиксом “@” подавляется эхо-печать отдельной команды пакетного файла, перед которой он установлен. Этот префикс позволяет более гибко управлять режимом эхо-печати команд пакетных файлов. При вводе команд с клавиатуры префикс @ системой не воспринимается и вызывает сообщение об ошибке:

Bad command or file name
(Неверная команда или имя файла).

Возможность использования префикса “@” появилась начиная с версии 3.3.

3. Для исключения дублирования выводимых сообщений при использовании команды ЕСНО в качестве оператора вывода в пакетных файлах предварительно необходимо отключать режим эхо-сопровождения, например

@ЕСНО OFF

ЕСНО ИДЕТ ОБРАБОТКА

4. Команда ЕСНО внутренняя.

Пример 25

C:\>echo of

C:\>

Пример 26

ECHO OFF

Пример 27

ECHO ON

.
. .
.

ECHO ВНИМАНИЕ!

Пример 28

@ECHO OFF

ECHO ВНИМАНИЕ!

Замечания к примерам. В примерах 25 и 26 показано отключение режима эхо-сопровождения с консоли и из пакетного файла. В первом случае это приведет к исчезновению системной подсказки (на экране останется только курсор). Однако команды по-прежнему будут восприниматься системой. Возобновить вывод подсказки можно путем выполнения команды ECHO ON. Во втором случае будет подавлена эхо-печать команд пакетного файла.

В примерах 27 и 28 приводится текст двух пакетных файлов. При выполнении первого из них сообщение ВНИМАНИЕ! будет выведено на экран дважды - вместе с командой ECHO и без нее. Для подавления дублирования вывода во второй пакетный файл включена дополнительная команда.

3.4. Команда GRAFTABL (команда установки/получения текущей таблицы кодов)

ОБЩИЕ СВЕДЕНИЯ. Команда GRAFTABL входит в состав MS-DOS и PC-DOS начиная с версии 3.0. Она загружает в оперативную память таблицу, содержащую унитарный образ произвольных символов (например, национальный алфавит, набор псевдографических символов и т.д.). Каждый символ при этом описывается матрицей размерностью 8x8 пикселей (элементов разложения), размещенной в восьми байтах памя-

ти. При каждом появлении кода символа матрица, содержащая унитарный образ заданного символа, переписывается программами BIOS в память видеоадаптера начиная с соответствующего адреса. Таким образом обеспечивается эффективная работа с символами в графическом режиме адаптера видеомонитора.

НАЗНАЧЕНИЕ. Команда GRAFTABL предназначена для переустановки текущей таблицы кодов, а также для вывода справочной информации и содержимого текущей таблицы кодов.

ОПИСАНИЕ. Команда GRAFTABL загружает в оперативную память шрифтов символы из расширенного набора ASCII с кодами 128-255 для обеспечения изображения этих символов в графическом режиме работы адаптера видеотерминала. При попытке повторной загрузки таблицы символов в оперативную память GRAFTABL выдает сообщение

Graphics characters already loaded
(Графические символы уже загружены).

Выйти из этой ситуации можно путем перезагрузки ОС.

Если в процессе работы GRAFTABL сталкивается с нарушением формата таблицы символов, на экран консоли выводится следующее сообщение об ошибке:

Non-standard version of Graphic Character Set Table
is already loaded

(Загружена нестандартная версия таблицы графических символов).

Из данной ситуации можно выйти, перезагрузив таблицы символов.

ФОРМАТЫ.

Ф о р м а т 1.

Выводит на консоль информацию о загруженных таблицах кодов.

[d:маршрут]GRAFTABL

Ф о р м а т 2.

Устанавливает таблицу кодов, заданную параметром *код*.

[d:маршрут]GRAFTABL код

Ф о р м а т 3.

Выводит на консоль информацию о загруженных таблицах кодов и допустимых параметрах команды GRAFTABL.

[d:маршрут]GRAFTABL ?

Ф о р м а т 4.

Выводит на консоль номер активной таблицы кодов.

[d:маршрут]GRAFTABL /STATUS

или

[d:маршрут]GRAFTABL /STA

Спецификация d:маршрут задает каталог, содержащий файл загрузочного модуля программы GRAFTABL. Она необходима, если этот каталог не текущий и не альтернативный.

В качестве параметра код могут быть использованы следующие значения:

- 437 - США (установлен по умолчанию);
- 850 - многоязыковая таблица;
- 860 - Португалия;
- 863 - Французская Канада;
- 865 - Дания, Норвегия.

Замечания.

1. При вызове из пакетного файла команда GRAFTABL возвращает пять кодов завершения, которые анализируются с помощью команды IF по условию ERRORLEVEL:

- 0 - успешное завершение;
- 1 - таблица символов уже загружена;
- 2 - ошибка доступа к файлу;
- 3 - неверный параметр;
- 4 - несовместимая версия DOS (требуется, чтобы номер версии программы GRAFTABL.COM точно совпадал с номером версии ядра ОС).

2. После загрузки GRAFTABL объем свободной оперативной памяти уменьшается примерно на 1,5 Кбайт.

3. Команда внешняя, выполняется программой, загрузочный модуль которой содержится в файле GRAFTABL.COM.

П р и м е р 29.

C:\>graftabl

No version of Graphic Character Set Table is already loaded.
USA version of Graphic Character Set Table has just been loaded.

C:\> graftabl /sta

USA version of Graphic Character Set Table is already loaded.

П р и м е р 30.

C:\> graftabl /sta

No version of Graphic Character Set Table is already loaded.

C:\>

П р и м е р 31.

C:\> graftabl ?

No version of Graphic Character Set Table is already loaded.

DOS command line parameters supported:

/STA	- Request Status only
?	- Display this summary of parameters
437	- USA Graphic Character Set
850	- Multilingual
860	- Portuguese Graphic Character Set
863	- Can. French Graphic Character Set
865	- Nordic Graphic Character Set

C:\>

Замечания к примерам. В примере 29 система выводит два сообщения:

1) таблица символов, предназначенная для работы в графическом режиме видеоадаптера, не загружена в оперативную память;

2) GRAFTABL загрузила таблицу символов USA - код 437, принятую по умолчанию.

Вторая команда GRAFTABL с ключом /STA подтверждает, что таблица символов USA загружена.

Пример 30 иллюстрирует действие ключа /STA. В отличие от предыдущей команды команда с ключом /STA не влияет на загрузку таблицы символов.

В примере 31 показан вывод справочной информации о команде GRAFTABL:

/STA	- запрос справки о текущем состоянии;
?	- высвечивает справочную информацию о команде GRAFTABL;
437	- графический набор символов USA;
850	- многоязыковая таблица;
860	- графический набор символов Португалии;
863	- графический набор символов Франц. Канады;
865	- графический набор символов Норвегии и Дании.

3.5. Команда KEYB (команда загрузки таблиц кодов клавиатуры)

ОБЩИЕ СВЕДЕНИЯ. Команда KEYB, реализованная начиная с версий PC-DOS 3.0 и MS-DOS 3.2, устанавливает драйвер клавиатуры, осуществляющий поддержку национальных алфавитов. Этот драйвер позволяет учесть все изобразительные особенности соответствующего алфавита и поставить системный набор символов в соответствии обозначениям национальной клавиатуры, разумеется, в рамках допустимых кодов стран. Другими словами, команда KEYB осуществляет поддержку клавиатур, не совпадающих со стандартной клавиатурой QWERTY.

Для реализации собственного специализированного режима работы клавиатуры пользователь может разработать свои таблицы кодов. При этом, чтобы управлять клавиатурой с помощью KEYB, необходимо точно соблюсти формат этих таблиц и интерфейс с драйвером. Альтернативным решением этой задачи может служить написание своего драйвера клавиатуры, но такое решение более трудоемко и менее мобильно.

НАЗНАЧЕНИЕ. Команда KEYB перезагружает таблицы кодов для клавиатуры. Обычно эта команда включается в состав пакетного файла AUTOEXEC.BAT.

ОПИСАНИЕ. В результате выполнения команды KEYB в резиденте системы устанавливается драйвер клавиатуры, кото-

рый шунтирует программу поддержки клавиатуры из ROM BIOS (базовой системы ввода-вывода, расположенной в ПЗУ) путем переустановки векторов прерываний. Этот драйвер будет преобразовывать SCAN-коды клавиш в коды символов соответственно специфицированной таблице символов.

Начиная с версии 3.3 таблицы кодировки символов для различных вариантов клавиатуры, учитывающие национальные особенности языков, сведены в специальный файл KEYBOARD.SYS (в более ранних версиях эта информация хранилась в резиденте системы в оперативной памяти).

После инициализации программы KEYB можно оперативно переключать два алфавита - США и национальный. Переключение на алфавит США производится после нажатия комбинации из трех клавиш - CTRL-ALT-F1, возврат к национальному алфавиту - клавиш CTRL-ALT-F2.

ФОРМАТ.

[d:*маршрут*]KEYB [xx[, [yyy], [*спец. файла*]]] [/ID:zzz]

Спецификация d:*маршрут* задает каталог, содержащий файл загрузочного модуля программы KEYB. Она необходима, если этот каталог не текущий и не альтернативный.

Параметр xx - двухсимвольный код для желаемой страны. Если код клавиатуры задан некорректно, программа KEYB выдает следующее сообщение:

Invalid keyboard code specified
(Специфицирован неверный код клавиатуры).

Параметр ууу - трехзначное число, обозначающее выбранную таблицу кодов. Если таблица кодов задана некорректно, программа KEYB выдает сообщение:

Invalid code page specified
(Специфицирована некорректная таблица кодов).

Спецификация спец.файла задает привод, маршрут, имя и расширение имени файла, содержащего программу-драйвер клавиатуры (по умолчанию - файл KEYBOARD.SYS в корневом каталоге текущего привода магнитных дисков). Если спецификация файла задана некорректно, программа KEYB выдает сообщение:

Bad or missing Keyboard Definition File
(Неверно задан или отсутствует файл определений клавиатуры).

Ключ /ID:zzz задает вариант идентификатора клавиатуры:

120	- Бельгия;
166/168	- Великобритания;
129	- Германия;
159	- Дания;
172	- Испания;
141/142	- Италия;
058	- Канада (французский);
171	- Латинская Америка;
143	- Нидерланды;
155	- Норвегия;
163	- Португалия;
103	- США;
153	- Финляндия;
120/189	- Франция;
000	- Швейцария (немецкий);
150	- Швейцария (французский);
153	- Швеция.

По умолчанию устанавливается значение 103 (США).

При вызове из пакетного файла команда KEYB возвращает семь кодов завершения, которые анализируются с помощью команды IF по условию ERRORLEVEL:

- 0 - успешное завершение;
- 1 - некорректная таблица кодов или синтаксис;
- 2 - ошибка доступа к магнитному диску;
- 3 - не удалось разместить таблицу клавиатуры в оперативной памяти;
- 4 - ошибка на консольном устройстве (на устройстве CON);
- 5 - запрошенная таблица кодов не привязана;
- 6 - в резидентной таблице клавиатуры не найдена таблица для выбранной таблицы кодов.

В версии 4.0 возвращается также код 7 - некорректная версия DOS.

Замечания.

1. Команда KEYB устанавливает альтернативный стандарт клавиатуры, если для соответствующей таблицы кодов выполнены команды:

MODE CON CODEPAGE PREPARE
MODE CON CODEPAGE SELECT

В противном случае при попытке выполнения команды KEYB будет выдаваться следующее сообщение:

Code page specified has not been prepared
(Специфицированная таблица кодов не привязана).

2. Ключ /ID реализован только в версии 4.0.

3. В версии 3.3 реализована поддержка только для 16 кодов стран.

4. Формирование кода завершения 3 сопровождается выводом следующего сообщения на экран консольного терминала:

Unable to create KEYB table in resident memory.
(Невозможно создать таблицу KEYB в резидентной памяти).

5. Для получения акцентов и умляутов в режиме клавиатуры, отличной от US, может быть использована так называемая потерянная клавиша (dead key). Нажатие этой клавиши не влечет за собой появление символа на экране, но видоизменяет последующий введенный символ.

6. Команда внешняя, выполняется программой, загрузочный модуль которой содержится в файле KEYB.COM.

П р и м е р 32.

```
C:\>keyb
Current keyboard code: GR Code page: 850
Current CON code page: 850
```

```
C:\>
```

П р и м е р 33.

```
C:\>keyb gr
```

```
C:\>
```

3.6. Команда MODE (установка характеристик внешних устройств)

ОБЩИЕ СВЕДЕНИЯ. Команда MODE реализована во всех версиях PC-DOS и MS-DOS, но от версии к версии существ-

венно наращивает свои возможности. Так, в версии 3.3 осуществляется настройка на таблицы кодов, а в версии 4.0 появилось несколько дополнительных форматов.

НАЗНАЧЕНИЕ. Команда **MODE** предназначена для установки режимов работы внешних устройств, инициализации таблиц кодов символов, а также для получения справочной информации о текущих параметрах настройки. Конкретное назначение для каждого формата команды расписано ниже.

ОПИСАНИЕ. Команда **MODE** - одна из самых мощных команд ОС. Загрузочный модуль программы, реализующей эту команду, занимает около 14,5 Кбайт для версии 3.3 и 23 Кбайт для версии 4.0.

ФОРМАТЫ ДЛЯ ВЕРСИИ DOS 3.3.

Ф о р м а т 1

[d:*маршрут*]**MODE** LPTx[:] [COLS],[LINES],[RETRY]

Формат 1 предназначен для управления режимами работы (характеристиками) принтеров, подключенных по параллельному интерфейсу. К таким характеристикам относятся количество символов в строке, количество строк на странице, режим обработки тайм-аута устройства. Назначения могут быть выполнены только в случае, если они задаются для принтера, совместимого по системе команд с принтерами IBM. Следует отметить, что, если принтер сбрасывается (инициализируется), значения вышеуказанных параметров устанавливаются по умолчанию в соответствии с алгоритмом работы устройства управления принтера (производится базовая инициализация принтера).

В этом и во всех последующих форматах версий 3.3 и 4.0 спецификация d:*маршрут* задает каталог, содержащий файл загрузочного модуля программы **MODE**. Спецификация необходима, если этот каталог не текущий и не альтернативный.

Параметр x задает номер параллельного порта, к которому подключен принтер:

- 1 - PRN или LPT1;
- 2 - LPT2;
- 3 - LPT3.

По умолчанию устанавливается LPT1.

Знак " : " в этом и во всех последующих форматах версий 3.3 и 4.0 - необязательный признак имени устройства.

Параметр COLS задает число символов в строке печатающего устройства - 80 или 132. По умолчанию устанавливается значение, соответствующее базовой инициализации используемого принтера (как правило, равное 80).

Параметр LINES задает число строк печати на дюйм (25,4 мм) бумаги. Это значение может составлять 6 или 8 строк/дюйм. По умолчанию устанавливается значение, соответствующее базовой инициализации используемого принтера (как правило, равное 6).

Параметр RETRY задает режим зацикливания запросов на печать в случае возникновения тайм-аута на устройстве. Ошибка квалифицируется как сбой по тайм-ауту, если в течение некоторого наперед заданного интервала времени устройство не обнаруживает никакой реакции на поступивший запрос. Этот интервал выбирается равным времени выполнения самой длительной операции обмена с устройством.

Замечания.

1. Не следует устанавливать параметр RETRY при работе в сетевом режиме.

2. При установленном параметре RETRY можно выйти из цикла подачи запросов на вывод информации на печать, вводя код CTRL-BREAK с клавиатуры.

Ф о р м а т 2

[d:маршрут]MODE LPTx[:]=COMy[:]

Формат 2 предназначен для инициализации режима обмена с принтерами через последовательные порты и используется при подключении принтера к компьютеру через последовательный порт коммуникаций.

Параметры x и y задают номера принтера и последовательного порта соответственно.

Ф о р м а т 3

[d:маршрут]MODE COMy[:]*скорость*[,паритет][,БИТД][,БИТС] [RETRY]

Формат 3 предназначен для управления характеристиками асинхронных последовательных коммуникационных портов, такими как скорость обмена, вид контроля по паритету, формат сообщений (количество информационных и служебных разрядов), режим обработки тайм-аутов внешнего устройства.

Параметр *у* задает номер последовательного коммуникационного порта - 1, 2, 3 или 4 (двоеточие - необязательный признак имени внешнего устройства).

Параметр *скорость* задает скорость обмена для соответствующего последовательного порта СОМу в бодах (1 Бод = 1 бит/с). Значение этого параметра может быть равным 110, 150, 300, 600, 1200, 2400, 4800, 9600 или 19200 Бод. Чем больше это значение, тем выше скорость обмена, но и тем выше вероятность снижения достоверности обмена. Скорость должна выбираться для каждого конкретного случая в зависимости от основных параметров системы:

- достоверность канала передачи данных между внешним устройством и контроллером, определяемая, в частности, интенсивностью обмена;

- тип внешнего устройства, подключенного через данный последовательный порт;

- тип соединительного кабеля;

- степень удаленности внешнего устройства и компьютера (контроллера);

- тип контроля по паритету;

- наличие в канале связи дополнительной аппаратуры передачи данных;

- степень помехозащищенности линии связи между внешним устройством и контроллером;

- характер, интенсивность и частотные характеристики электромагнитных помех.

Целесообразно устанавливать максимальную скорость, обеспечивающую необходимый уровень достоверности передачи данных при обмене.

Параметр *паритет* задает вид контроля паритета для соответствующего порта СОМ. Разрешены следующие значения:

Е (even) - контроль по четности;

О (odd) - контроль по нечетности;

Н (none) - отсутствие контроля по паритету.

Значение Е (even) устанавливается по умолчанию.

Параметр БИТД задает число используемых информационных разрядов, равное 7 или 8 (в зависимости от разновидности символического кода). По умолчанию - число 7.

Параметр БИТС устанавливает число используемых служебных разрядов (стоп-бит) - 1 или 2. По умолчанию оно вы-

бирается в зависимости от текущей скорости передачи: для скорости 110 Бод - 1, для остальных скоростей - 2.

Параметр **RETRY** задает режим зацикливания запросов на обмен информацией с коммуникационным портом при возникновении сбоя по тайм-ауту.

Ф о р м а т 4

[d:маршрут]MODE *тип экрана*

Формат 4 предназначен для установки режима работы адаптера видеомонитора.

Параметр *тип экрана* задает режим отображения. Разрешены значения:

40 - переход в режим 40 символов в строке;

80 - переход в режим 80 символов в строке;

BW40 - переход в режим 40 символов в строке адаптера CGA с запрещением цвета;

BW80 - переход в режим 80 символов в строке адаптера CGA с запрещением цвета;

CO40 - переход в режим 40 символов в строке адаптера CGA с разрешением цвета;

CO80 - переход в режим 80 символов в строке адаптера CGA с разрешением цвета;

MONO - переход в режим монохромного адаптера с постоянным размером строки, равным 80 символам.

Ф о р м а т 5

[d:маршрут]MODE [*тип экрана*], *коррекция* [,T]

Формат 5 предназначен для установки левой и правой границ экрана.

Параметр *тип экрана* аналогичен соответствующему параметру формата 4.

Параметр *коррекция* задает направление горизонтальной юстировки экрана. Разрешены два значения: L - влево и R - вправо.

Параметр T выводит на экран тестовый пример для обеспечения возможности проверки правильности установки параметров в процессе юстировки.

Ф о р м а т 6

[d:маршрут]MODE устройство[:] CODE PAGE
PREPARE = ((xxx)спец.файла)

Формат 6 предназначен для привязки таблиц кодов символов к внешним устройствам.

Допускается использование следующих сокращенных ключевых слов:

[d:маршрут]MODE устройство[:] CP PREP =
((XXX)спец.файла)

В качестве параметра *устройство* задается имя любого из следующих внешних устройств, для которых реализована возможность работы в режиме перезагружаемых таблиц кодов:

CON - консольный терминал;

PRN, LPT1, LPT2, LPT3 - принтеры, подключенные к параллельному или последовательному порту COM1, COM2, COM3, но переназначенные с помощью формата 2 команды MODE.

Параметр XXX задает код привязываемой таблицы кодов, который может принимать одно из следующих пяти значений:

437 - США;

850 - многоязыковая таблица;

860 - Португалия;

863 - Французская Канада;

865 - Дания, Норвегия.

Параметр *спец.файла* задает спецификацию файла, содержащего таблицу кодов, которую необходимо загрузить в оперативную память. В качестве умолчания для имени привода и каталога используются текущие привод и каталог; в качестве умолчания для типа файла - .CPI - Code Pages Information (информация о таблицах кодов). В состав версии 3.3 входят следующие файлы, содержащие таблицы кодов:

EGA.CPI - для адаптеров EGA и VGA;

LCD.CPI - для ЖКИ-дисплея компьютеров, совместимых с IBM PC Convertible;

4201.CPI - для принтеров, совместимых с принтерами IBM Proprinter 4201, 4202, 4207;

5202.CPI - для принтеров, совместимых с принтерами IBM Quietwriter III 5202.

Ф о р м а т 7

[d:маршрут]MODE устройство[:] CODE PAGE
SELECT = XXX

Формат 7 предназначен для выбора одной из привязанных таблиц кодов.

Допускается использование сокращенных ключевых слов:

[d:маршрут]MODE устройство[:] CP SEL = XXX

Параметры *устройство* и XXX аналогичны соответствующим параметрам предыдущего формата.

Таблица кодов может быть выбрана (активизирована) только после ее привязки с помощью команды MODE предыдущего формата.

Ф о р м а т 8

[d:маршрут]MODE устройство[:]CODE PAGE[/STATUS]

Формат 8 предназначен для получения справочной информации о привязанных таблицах кодов внешнего устройства и об активной таблице кодов.

Допускается использование сокращенных ключевых слов:

[d:маршрут]MODE устройство CP[:] [/STA]

Параметр *устройство* аналогичен соответствующему параметру предыдущего формата.

Ключ /STA - необязательный ключ-указатель запроса на вывод справочной информации. Он позволяет получить номер активной таблицы кодов, а также номера всех привязанных к заданному устройству таблиц.

Ф о р м а т 9

[d:маршрут]MODE устройство[:] CODE PAGE REFRESH

Формат 9 восстанавливает привязанную таблицу кодов, если ее копия, применяемая при работе с устройством, оказалась запорченной.

Допускается использование сокращенных ключевых слов:

[d:маршрут]MODE устройство[:] CP REF

ФОРМАТЫ ДЛЯ ВЕРСИИ DOS 4.0.

Ф о р м а т 1

[d:маршрут]MODE LPTx[:][COLS = A][LINES = B]
[RETRY = C]

Формат 1 предназначен для управления режимами работы принтеров, такими как число символов в строке, число строк на странице, режим обработки тайм-аута устройства.

Допускается использование сокращенного формата:

[d:маршрут]MODE LPTx[:] [A],[B],[C]

Параметры x, A, B полностью эквивалентны параметрам COLS, LINES, RETRY формата 1 версии 3.3, описанным выше.

Параметр C задает режим заикливания запросов на вывод информации при обработке сбоев по тайм-ауту и может принимать следующие значения:

E - возвращает ошибку контроля состояния порта;

B - возвращает признак занятости порта;

R - возвращает признак готовности;

NONE - операция повтора запросов не выполняется.

По умолчанию устанавливается значение E.

Ф о р м а т 2

[d:маршрут]MODE LPTx[:] = COMy[:]

Формат 2 используется для инициализации режима обмена с принтерами через последовательные порты и полностью эквивалентен формату 2 версии 3.3.

Ф о р м а т 3

[d:маршрут]MODE COMy[:] BAUD = скорость[PARITY = паритет][DATA = БИТД][STOP = БИТС] [RETRY = C]

Формат 3 предназначен для управления характеристиками последовательных портов, таких как скорость обмена, вид контроля по паритету, формат сообщений (количество инфор-

мационных и служебных разрядов), режим обработки таймаутов внешнего устройства.

Допускается использование сокращенного формата

[d:маршрут]MODE COMy[:] скорость[,паритет][,БИТД] [,БИТС][,С]

Параметры *y*, *скорость*, *паритет*, БИТД и БИТС полностью аналогичны соответствующим параметрам формата 1 версии 3.3, описанным выше.

Параметр *С* задает режим зацикливания запросов при обработке сбоев по тайм-ауту и может принимать следующие значения:

Е - возвращает ошибку контроля состояния порта;

В - возвращает признак занятости порта;

Р - возвращает признак готовности;

NONE - операция повтора запросов не выполняется.

По умолчанию устанавливается значение Е.

Ф о р м а т 4

[d:маршрут]MODE *тип экрана*, N

Формат 4 предназначен для установки режима работы адаптера видеомонитора.

Параметр *тип экрана* полностью аналогичен соответствующему параметру формата 4 версии 3.3, описанному выше.

Параметр *N* специфицирует количество строк экрана и может принимать значения 25, 43 или 50. Задание любого другого значения вызывает выдачу сообщения об ошибке:

Invalid parameter - lines = xx

(Неверный параметр lines = xx, где xx - количество строк, специфицированное в командной строке).

Если установка параметра на данной конфигурации невозможна, система выдает сообщение:

Function not supported on this computer

(На данном компьютере функция не поддерживается).

Следует отметить, что значения 43 и 50 совместимы не со всеми типами видеоадаптеров и мониторов. Установка этих параметров может вызывать десинхронизацию между размерами страницы памяти видеоадаптера и экрана монитора. В та-

ких случаях текущая позиция (маркер) смещается за пределы экрана.

Ф о р м а т 5

[d:маршрут]MODE [тип экрана], коррекция [,T]

Формат 5 предназначен для установки левой и правой границ экрана.

Параметры *тип экрана*, *коррекция* и T полностью эквивалентны соответствующим параметрам формата 5 версии 3.3, описанным выше.

Ф о р м а т 6

[d:маршрут]MODE CON[:] [COLS = M][LINES = N]

Формат 6 задает нужное количество строк символов и символов в строке экрана консольного терминала.

Параметр M специфицирует количество символов в строке экрана. Допустимые значения - 40 и 80.

Значение параметра N специфицирует количество строк экрана и задается аналогично соответствующему параметру формата 4.

Ф о р м а т 7

[d:маршрут]MODE CON[:] RATE = r DELAY = d

Формат 7 устанавливает нужные временные параметры процесса сканирования клавиш клавиатуры.

Параметр r задает количество циклов опроса клавиатуры в секунду. Это значение может устанавливаться в пределах 1...32. Увеличение значения параметра улучшает время реакции системы. Соответственно увеличиваются аппаратные издержки на сканирование.

Установка максимального значения параметра r дает особенно заметный эффект при многократном вводе одного и того же кода длительным нажатием клавиши (например, при перемещении маркера по экрану в редакторе текста).

Параметр d задает код величины задержки с момента нажатия клавиши, определяющей время, после которого начнут восприниматься повторные нажатия этой клавиши. Допустима установка следующих значений величины задержки: 1) 250 мс, 2) 500 мс, 3) 750 мс, 4) 1 с.

Установка минимального значения параметра *d* дает существенный эффект при использовании многократного нажатия клавиш (например, при перемещении маркера по экрану в редакторе текста).

Ф о р м а т 8

[*d:маршрут*]MODE *устройство[:]* CODE PAGE
PREPARE = ((*xxx*)*спец.файла*)

Формат 8 позволяет привязать таблицу кодов символов к внешнему устройству.

Допускается использование сокращенных ключевых слов:

[*d:маршрут*]MODE *устройство[:]* CP PREP =
((*xxx*)*спец.файла*)

Параметр *спец.файла* задает спецификацию файла, содержащего таблицу кодов, которую необходимо загрузить в оперативную память. По умолчанию для имени привода и каталога используются текущие привод и каталог; для типа файла - .CPI - Code Pages Information (информация о таблицах кодов). В состав версии 3.3 входят следующие файлы, содержащие таблицы символов:

EGA.CPI	- для адаптеров EGA и VGA;
LCD.CPI	- для ЖКИ-дисплея компьютеров, совместимых с IBM PC Convertible.
4201.CPI	- для принтеров, совместимых с принтерами IBM Proprinter 4201, 4202, 4207;
4208.CPI	- для принтеров, совместимых с принтерами IBM Proprinter 4208;
5202.CPI	- для принтеров, совместимых с принтерами IBM Quietwriter III 5202.

Параметры *устройство* и *xxx* полностью эквивалентны соответствующим параметрам формата 6 версии 3.3, описаным выше.

Ф о р м а т 9

[*d:маршрут*]MODE *устройство[:]* CODE PAGE
PREPARE = ((CPList)*спец.файла*)

Формат 9 позволяет привязать набор таблиц кодов символов к внешнему устройству.

Допускается использование сокращенных ключевых слов:

[d:маршрут]MODE устройство CP PREP =
((CPList)спец.файла)

Параметр *устройство* полностью эквивалентен соответствующему параметру формата 6 версии 3.3, описанному выше.

Параметр *спец.файла* аналогичен параметру *спец.файла*, описанному в предыдущем формате.

Параметр CPList задает список номеров таблиц кодов.

Ф о р м а т 10

[d:маршрут]MODE устройство[:] CODE PAGE
SELECT = xxx

Формат 10 предназначен для выбора одной из привязанных таблиц кодов.

Допускается использование сокращенных ключевых слов

[d:маршрут]MODE устройство[:] CP SEL = xxx

Формат 10 полностью эквивалентен формату 7 версии 3.3, приведенному выше.

Ф о р м а т 11

[d:маршрут]MODEустройство[:]CODE PAGE [/STATUS]

Формат 11 предназначен для получения справочной информации о таблицах кодов внешнего устройства (привязанных и активной таблицах кодов).

Допускается использование сокращенных ключевых слов

[d:маршрут]MODE устройство[:] CP [/STA]

Формат 11 полностью эквивалентен формату 8 версии 3.3, приведенному выше.

Ф о р м а т 12

[d:маршрут]MODE устройство[:] CODE PAGE REFRESH

Формат 12 позволяет восстановить привязанную таблицу кодов, если ее копия, используемая при работе с устройством, оказалась испорченной.

Допускается применение сокращенных ключевых слов:

[d:маршрут]MODE устройство[:] CP REF

Формат 12 полностью эквивалентен формату 9 версии 3.3, приведенному выше.

Ф о р м а т 13

[d:маршрут]MODE

Формат 13 позволяет получить справочную информацию о параметрах внешних устройств, имеющихся в конфигурации, и о загруженных таблицах кодов. Это касается устройств CON, LPT1, LPT2, LPT3, COM2, COM3, COM4, переустановка которых возможна с помощью команды MODE.

Замечания.

1. В версии 3.3 не реализованы следующие возможности команды MODE версии 4.0:

- в форматах 1 и 3 - значения E, B, R, NONE для параметра RETRY = C;
- в формате 4 - параметр N, специфицирующий количество строк экрана;
- кроме того, отсутствуют форматы 6, 7, 9 и 13;
- в формате 8 нет поддержки для принтеров, совместимых с принтерами IBM Proprinter 4208 (файл 4208.CPI, содержащий соответствующие таблицы кодов).

2. Для реализации функций управления таблицами кодов в системе должен быть установлен драйвер DISPLAY.SYS с помощью команды DEVICE, размещенной в файле конфигурирования системы CONFIG.SYS, например

DEVICE = DISPLAY.SYS CON = (EGA)

3. Команда внешняя, выполняется программой, загрузочный модуль которой содержится в файле MODE.COM.

П р и м е р 34.

C:\>ver

IBM DOS Version 4.00

C:\>

C:\>mode

Status for device LPT1:

LPT1: not rerouted
RETRY = NONE

Code page operation not supported on this device

Status for device LPT2:

LPT2: not rerouted

Status for device LPT3:

LPT3: not rerouted

Status for device CON:

COLUMNS = 80

LINES = 25

Code page operation not supported on this device

C:\>

П р и м е р 35.

C:\>ver

MS-DOS Version 3.30

C:\>mode con cp /sta

No code page has been SELECTED

prepared code pages:

Code page not prepared

Device CON not prepared

MODE Status Code page function completed

C:\>mode con cp prep = (437)ega.cpi

MODE Prepare Code page function completed

C:\>mode con cp /sta

No code page has been SELECTED

prepared code pages:

Code page 437

MODE Status Code page function completed

C:\>mode con cp prep = (850)ega.cpi

MODE Prepare Code page function completed

C:\>mode con cp /sta

No code page has been SELECTED

prepared code pages:

Code page 850

MODE Status Code page function completed

C:\>mode con cp sel = 850

MODE Select Code page function completed

C:\>mode con cp /sta

Active code page for device CON is 850

prepared code pages:

Code page 850

MODE Status Code page function completed

C:\>nlsfunc

C:\>chcp 850

C:\>chcp

Active code page: 850

C:\>keyb gr

C:\>keyb

Current keyboard code: GR code page: 850

Current CON code page: 850

C:\>

П р и м е р 36.

C:\>ver

MS-DOS Version 4.01

C:\>keyb gr

Code page specified has not been prepared

C:\>mode con cp prep = ((850)ega.cpi)

MODE prepare code page function completed

C:\>mode

Status for device LPT1:

LPT1: not rerouted

RETRY = NONE

Code page operation not supported on this device

Status for device LPT2:

LPT2: not rerouted

Status for device LPT3:

LPT3: not rerouted

Status for device CON:

COLUMNS = 80

LINES = 25

No code page has been selected

Prepared code pages:

code page 850

MODE status code page function completed

Status for device COM1:

RETRY = NONE

C:\>keyb gr

Code page specified has not been prepared

C:\>mode con cp sel = 850

MODE select code page function completed

C:\>mode

Status for device LPT1:

LPT1: not rerouted

RETRY = NONE

Code page operation not supported on this device

Status for device LPT2:

LPT2: not rerouted

Status for device LPT3:

LPT3: not rerouted

Status for device CON:

COLUMNS = 80

LINES = 25

Active code page for device CON is 850

Prepared code pages:

code page 850

C:\>keyb gr

C:\>keyb

Current keyboard code: GR code page: 850

Current CON code page: 850

C:\>

П р и м е р 37.

C:\>ver

MS-DOS Version 4.01

C:\>mode con delay = 1

RATE and DELAY must be specified together

C:\>mode con rate = 32delay = 1

Parameter format not correct - rate = 32delay = 4

C:\>mode con rate = 32 delay = 1

C:\>

Замечания к примерам.

Примеры команды **MODE** даны для версий MS-DOS 4.01, MS-DOS 3.3 и IBM-DOS 4.0. Для идентификации версий системы в начале каждого примера показан результат выполнения команды **VER**.

В примере 34 приведена справка об исходном состоянии внешних устройств, полученная с помощью формата 13 версии 4.0 до выполнения каких бы то ни было установок с помощью команды **MODE**.

Пример 35 представляет собой процедуру инициализации многоязыковой таблицы кодов (850) и перевод клавиатуры в режим немецкого стандарта. Для операции **PREPARE** приводятся результаты выполнения последовательности команд (показано, что действует только последняя привязка). Изменение состояния до и после выполнения каждой операции отображается с помощью справочных команд.

В примере 36 показана минимальная процедура перевода клавиатуры в режим немецкого стандарта. На каждом шаге отражена реакция системы на преждевременное выполнение финишной операции.

В примере 37 описана процедура перевода клавиатуры в режим минимального времени реакции на нажатие клавиши. Предварительно приводятся две команды, заданные ошибочно, и реакция системы на них.

3.7. Команда NLSFUNC (установка режима смены таблиц кодов)

ОБЩИЕ СВЕДЕНИЯ. Команда NLSFUNC (National Language Support FUNCTIONS - функции поддержки национальных языков) впервые реализована в версии 3.3.

Команда NLSFUNC представляет собой одно из звеньев системы программной поддержки национальных алфавитов. Общая укрупненная схема системных средств, предназначенных для обеспечения такой поддержки, имеет следующий вид:

= = » Файл конфигурирования системы
CONFIG.SYS

КОНФИГУРИРОВАНИЕ = = » Файл автонастройки системы
AUTOEXEC.BAT

= = » Команда SELECT

= = » Команда NLSFUNC

ИНИЦИАЛИЗАЦИЯ = = » Команда MODE (PREPARE, SELECT)

= = » Команда KEYB

= = » Команда MODE (STATUS, REFRESH)

РАБОЧИЕ ПРОЦЕДУРЫ = = » Команда CHCP

= = » Команда GRAFTABL

НАЗНАЧЕНИЕ. Команда NLSFUNC обеспечивает возможность использования информации, специфичной для страны и расположенной во внешней памяти компьютера. По этой команде информация, специфичная для страны, загружается в оперативную память (структуру блока информации, специфичной для страны, см. в описании команды COUNTRY).

ОПИСАНИЕ. Команда NLSFUNC поддерживает режим управления сменой таблиц кодов с помощью команды CHCP (см. описание команд MODE и CHCP).

Программа NLSFUNC загружает всю необходимую специфическую информацию о странах. Пока команда NLSFUNC не будет выполнена, доступ к таблицам кодов, отличным от стандартной таблицы 437, невозможен.

В процессе выполнения команды NLSFUNC формирует резидент в оперативной памяти. NLSFUNC может быть удален только путем перезагрузки ОС. Попытка повторного выполнения команды приведет к выводу сообщения об ошибке:

NLSFUNC already installed
(NLSFUNC уже установлена).

ФОРМАТ.

[*d:маршрут*]**NLSFUNC** [*спец.файла*]

Спецификация *d:маршрут* задает каталог, содержащий файл загрузочного модуля программы NLSFUNC. Она необходима, если этот каталог не текущий и не альтернативный.

Спецификация *спец.файла* задает файл, содержащий информацию о стране. В качестве имени файла по умолчанию принимается имя файла, специфицированное в команде COUNTRY, содержащейся в файле CONFIG.SYS. Если команда COUNTRY отсутствует, то именем по умолчанию становится COUNTRY.SYS.

Замечания.

1. После выполнения команды NLSFUNC размер свободной оперативной памяти уменьшается примерно на 2,5 Кбайт для версий 3.3 и на 3 байт для версий 4.0.

2. Команда внешняя. Выполняется программой, загрузочный модуль которой содержится в файле NLSFUNC.EXE.

П р и м е р 38.

C:\>chkdsk

Volume 4016SHELL created 10-01-1990 4:33p
Volume Serial Number is 275D-14CA

33462272 bytes total disk space
73728 bytes in 3 hidden files
63488 bytes in 28 directories
14958592 bytes in 532 user files
18366464 bytes available on disk

2048 bytes in each allocation unit
16339 total allocation units on disk
8968 available allocation units on disk

655360 total bytes memory
554096 bytes free

C:\>nlsfunc

C:\>chkdsk

Volume 4016SHELL created 10-01-1990 4:33p

Volume Serial Number is 275D-14CA

33462272 bytes total disk space

73728 bytes in 3 hidden files

63488 bytes in 28 directories

14960640 bytes in 532 user files

18364416 bytes available on disk

2048 bytes in each allocation unit

16339 total allocation units on disk

8967 available allocation units on disk

655360 total bytes memory

551136 bytes free

C:\>

Из справок, полученных командами CHKDSK до и после установки NLSFUNC, видно, что объем резидента системы возрастает ориентировочно на 3 Кбайт.

3.8. Команда PROMPT (установка формата системной подсказки)

ОБЩИЕ СВЕДЕНИЯ. Команда PROMPT реализует простой язык, описывающий произвольный формат системной подсказки. При этом пользователю предоставляются широкие возможности по управлению форматом подсказки. Кроме того, команда PROMPT имеет ряд возможностей по управлению экраном и клавиатурой, не связанных с непосредственным формированием системной подсказки. Основные из них:

- перенастройка произвольных клавиш клавиатуры на ввод произвольных последовательностей символов;
- выбор цвета основного и фоновых изображений, а также вспомогательных сообщений, выводимых на экран (например, времени);
- формирование распределенных сообщений, расположенных в произвольных несвязных областях экрана;

- управление произвольными перемещениями курсора;
- вывод системной даты и времени;
- возможность вывода номера версии ОС.

НАЗНАЧЕНИЕ. Команда **PROMPT** предназначена для программирования формата системной подсказки и настройки клавиатуры и экрана.

ОПИСАНИЕ. Язык, реализуемый командой **PROMPT**, состоит из двух частей - постоянной, представляющей собой набор команд для описания наперед заданных элементов системной подсказки, и переменной, позволяющей инициировать выполнение ESC-последовательностей драйвером клавиатуры (в частности, расширенным драйвером **ANSI.SYS**, поставляемым с ОС). Программа, описанная командой **PROMPT**, выполняется после очередной команды пользователя. Команда **PROMPT** реализуется с помощью драйверов и библиотечных подпрограмм системной поддержки экрана и клавиатуры консольного терминала.

ФОРМАТ.

PROMPT [цепочка символов]

Цепочка символов - это произвольная совокупность (конкатенация) двухсимвольных сочетаний из нижеприведенного перечня.

- \$t** - вывод текущего системного времени.
- \$d** - вывод текущей системной даты.
- \$p** - вывод текущего маршрута.
- \$v** - вывод номера версии ОС.
- \$n** - вывод имени привода по умолчанию.
- \$g** - вывод символа ">".
- \$l** - вывод символа "<".
- \$b** - вывод символа "|".
- \$q** - вывод символа "=".
- \$a** - вывод символа "&".
- \$h** - выполнение функции <BACKSPACE> (забой).
- \$_** - вывод символов <RETURN> (возврат каретки) и <Line Feed> (перевод строки).

\$e - вывод кода <ESCAPE> (переход - с одного кода на другой), позволяющего управлять экраном с помощью так называемых ESC-последовательностей.

Используя ESC-последовательности под управлением стандартного системного загружаемого драйвера ANSI.SYS, можно выполнить следующие функции:

\$e[x;yH - перемещение курсора в позицию x строки y; если параметры не указаны, то курсор устанавливается в позицию HOME (1,1).

\$e[x;yf - перемещение курсора в позицию x строки y; если параметры не указаны, то курсор устанавливается в позицию HOME (1,1).

\$e[yA - перемещение курсора на y строк вверх без изменения позиции в строке; если курсор на верхней строке, то команда игнорируется.

\$e[yB - перемещение курсора на y строк вниз без изменения позиции в строке; если курсор на нижней строке, то команда игнорируется.

\$e[xC - перемещение курсора на x позиций вправо без изменения номера строки; если курсор в крайней правой позиции, то команда игнорируется.

\$e[xD - перемещение курсора на x позиций влево без изменения номера строки; если курсор в крайней правой позиции, то команда игнорируется.

\$e[6n - выдача (эмуляция) текущих координат (номера строки и столбца) курсора на стандартное устройство ввода. формат - **\$e[x,yR** .

\$e[s - запоминание текущей позиции курсора, эта позиция может быть восстановлена с помощью последовательности **\$e[u** .

\$e[u - восстановление позиции курсора, запомненной с помощью последовательности **\$e[s** .

\$e[2J - очистка экрана и перевод курсора в левый верхний угол экрана (позицию HOME).

\$e[K - удаление строки (от позиции, занимаемой курсором, до конца строки).

\$e[xl; - выполнение набора графических функций:
...;xNm

- 0 - сброс всех атрибутов;
- 1 - установка режима полужирного шрифта;
- 2 - установка режима тусклого шрифта;
- 3 - установка режима вывода курсивом;

- 5 - установка режима мерцания;
- 6 - установка режима мерцания с высокой частотой;
- 7 - установка режима реверсивного отображения;
- 8 - установка скрытого режима;
- 30 - установка черного цвета выводных сообщений;
- 31 - установка красного цвета выводных сообщений;
- 32 - установка зеленого цвета выводных сообщений;
- 33 - установка желтого цвета выводных сообщений;
- 34 - установка синего цвета выводных сообщений;
- 35 - установка малинового цвета выводных сообщений;
- 36 - установка голубого цвета выводных сообщений;
- 37 - установка белого цвета выводных сообщений;
- 40 - установка черного цвета фона;
- 41 - установка красного цвета фона;
- 42 - установка зеленого цвета фона;
- 43 - установка желтого цвета фона;
- 44 - установка синего цвета фона;
- 45 - установка малинового цвета фона;
- 46 - установка голубого цвета фона;
- 47 - установка белого цвета фона;
- 48 - установка нижнего индекса фона;
- 49 - установка верхнего индекса фона.

\$e[= xh - установка режима экрана x:

- 0 - символьный черно-белый 40x25 символов;
- 1 - символьный цветной 40x25 символов;
- 2 - символьный черно-белый 80x25 символов;
- 3 - символьный цветной 80x25 символов;
- 4 - цветной графический 320x200 пикселей;
- 5 - графический черно-белый 320x200 пикселей;
- 6 - графический монохромный 640x200 пикселей;
- 7 - переходы при достижении конца каждой строки;
- 14 - графический цветной 640x200 пикселей;
- 15 - графический монохромный 640x350 пикселей;
- 16 - графический цветной 640x350 пикселей;
- 17 - графический цветной 640x480 пикселей;
- 18 - графический цветной 640x480 пикселей;
- 19 - графический цветной 320x200 пикселей;

\$e[= x1 - сброс режима x.

\$e[y;z;...p - поставить в соответствие клавише с кодом y строку символов z. Допустимы следующие коды клавиш:

Название клавиши	Отдельное нажатие клавиши	Нажатие вместе с клавишей		
		SHIFT	CTRL	ALT
A	97	65	1	0;30
B	98	66	2	0;48
C	99	67	3	0;46
D	100	68	4	0;32
E	101	69	5	0;18
F	102	70	6	0;33
G	103	71	7	0;34
H	104	72	8	0;35
I	105	73	9	0;23
J	106	74	10	0;36
K	107	75	11	0;37
L	108	76	12	0;38
M	109	77	13	0;50
N	110	78	14	0;49
O	111	79	15	0;24
P	112	80	16	0;25
Q	113	81	17	0;16
R	114	82	18	0;19
S	115	83	19	0;31
T	116	84	20	0;32
U	117	85	21	0;22
V	118	86	22	0;47
W	119	87	23	0;17
X	120	88	24	0;45
Y	121	89	25	0;21
Z	122	90	26	0;44
0	48	41	--	0;129
1	49	33	--	0;120
2	50	64	--	-----
3	51	35	--	0;122
4	52	36	--	0;123
5	53	37	--	0;124
6	54	94	--	0;125
7	55	38	--	0;126
8	56	42	--	0;127
9	57	40	--	0;128
=	61	43	--	0;131
Tab	9	0;15	--	-----
Null	0;3	----	--	-----
F1	0;59	0;84	0;94	0;104
F2	0;60	0;85	0;95	0;105
F3	0;61	0;86	0;96	0;106
F4	0;62	0;87	0;97	0;107
F5	0;63	0;88	0;98	0;108
F6	0;64	0;89	0;99	0;109
F7	0;65	0;90	0;100	0;110
F8	0;66	0;91	0;101	0;111

Окончание табл.

F9	0;67	0;92	0;102	0;112
F10	0;68	0;93	0;103	0;113
F11	0;133	0;135	0;137	0;139
F12	0;134	0;136	0;138	0;140
Home	0;71	55	0;119	----
СтрВвех	0;72	56	----	----
Pg Up	0;73	57	0;132	----
СтрВлево	0;75	52	0;115	----
СтрВверх	0;77	54	0;116	----
End	0;79	49	0;117	----
Pg Dn	0;81	51	0;118	----
Ins	0;82	48	----	----
Del	0;83	46	----	----
PrtSc	----	--	0;114	----

Замечания.

1. ESC-последовательности используются только, если при конфигурировании системы под управлением файла CONFIG.SYS была выполнена команда установки расширенного драйвера клавиатуры ANSI.SYS:

DEVICE = ANSI.SYS

2. Следует помнить, что в конструкциях PROMPT строчные и прописные буквы латинского алфавита интерпретируются по-разному и могут иметь различное назначение при описании параметров.

3. Команда PROMPT внутренняя.

П р и м е р 39.

prompt

П р и м е р 40.

prompt \$p\$g

П р и м е р 41.

prompt \$e[s\$e[1;60f\$e[37;42;1m \$t\$h\$h\$h\$h\$h\$h\$h \$d\$e[u\$e[35m
\$e[44m \$p\$q\$g

П р и м е р 42.

prompt \$e[0;68;"S";13p\$e[13;"C";13p\$e[s\$e[1;60H\$e[37;42;
1m \$t \$h\$h\$h\$h\$h\$h\$h \$d\$e[u\$e[35m\$e[44m \$p\$q\$g

Файл C.BAT

```
PROMPT $e[0;30;44;1m $e[2J
CTTY NUL
```

Файл S.BAT

```
@echo off
prompt $e[13;"";13p$e[s$e[1;60H$e[37;42;1m $t$h$h$h$h$h$h$h
$d $e[u$e[35m$e[44m $p$q$g
```

В примере 39 описывается установка формата подсказки по умолчанию.

В примере 40 приводится самый распространенный в обиходе (и один из самых простых) формат системной подсказки, который формируется путем вывода в первую позицию строки имени привода по умолчанию и маршрута текущего каталога (\$p), а также знака больше ">".

В примере 41 приводится более сложная программа вывода системной подсказки. Эта программа начинается с обращения к драйверу ANSI.SYS с помощью ESC-последовательности \$e[s. Такая команда вызывает запоминание текущей позиции курсора (номера строки и номера позиции в строке) во внутренних структурах данных драйвера. Следующая команда (\$e[1;60f) переводит курсор в шестидесятую позицию первой строки, подготовив таким образом экран для вывода текущей даты и времени. Далее следует команда установки цвета основного и фоновых изображений \$e[37;42;1m (белый текст на зеленом фоне).

Следующая команда \$t выполняется уже непосредственно программой PROMPT и выводит на экран консольный терминал текущего системного времени в соответствии с кодом формата, установленным в блоке описания информации, специфичной для страны (структуру блока информации, специфичной для страны, см. в описании команды COUNTRY).

Так как информация о секундах и долях секунды при повседневном применении не имеет значения, она забивается шестью командами забоя (\$h). После этого на экран выводится один пробел, а затем системная дата (\$d), после чего следует обращение к драйверу ANSI.SYS для восстановления исходной позиции курсора, которая была сохранена в начале выполнения программы (команда \$e[u).

Далее переустанавливается цвет основного и фоновых изображений уже двумя командами - \$e[35m и \$e[44m (малиновый текст на синем фоне). Использование отдельных ко-

манд для управления цветом может впоследствии дать некоторые удобства при редактировании программы вывода системной подсказки. В заключение на экран выводится пробел, имя привода по умолчанию и маршрут текущего каталога (\$p), знак равенства (\$q) и знак больше (\$g).

Пример 42 - это программа, которая кроме функции вывода собственно системной подсказки, даты и времени, описанных в предыдущем примере, выполняет также функцию защиты системы от несанкционированного доступа после загрузки ОС с жесткого диска, для чего команда PROMPT использует два пакетных файла - C.BAT и S.BAT, текст которых приведен в этом же примере. Рассмотрим механизм действия этой защиты, не расписывая часть команды PROMPT, аналогичную команде, разобранный в предыдущем примере.

Программа начинается с команды, переназначающей клавишу F10, которая в дальнейшем будет использована в качестве ключа входа в систему \$e[0;68;"S";13p. После выполнения этой команды по нажатию клавиши F10 будет запускаться пакетный файл S.BAT. Этот файл содержит команду PROMPT, которая восстанавливает стандартное назначение клавиш клавиатуры и устанавливает формат подсказки, аналогичный формату, описанному в команде PROMPT из файла AUTOEXEC.BAT (первая команда PROMPT в примере).

Затем с помощью команды \$e[13;"C";13p переназначается клавиша ENTER, которая начинает выполнять функцию запуска пакетного файла C.BAT. Файл C.BAT состоит из двух команд, первая устанавливает режим наименьшего контраста изображения, а вторая - переназначает системную консоль на ноль-устройство NUL. После выполнения этого файла дальнейшая работа с системой становится, естественно, невозможной (см. описание команды CTTY).

Таким образом, устанавливая произвольные коды вместо кода клавиши F10, можно организовывать вход в систему по нажатию произвольной клавиши.

Внешне действие такой защиты проявляется следующим образом. После загрузки системы ввод практически любой команды вызывает сообщение об ошибке:

Bad command or file name
(Неверная команда или имя файла).

Это происходит потому, что к каждой командной строке приписывается буква C, что, в свою очередь, делает команду или имя файла нераспознаваемыми. Разумеется, что в этом случае пакетный файл C.BAT выполняться не будет.

Только отдельное нажатие клавиши ENTER приведет к зависанию системы.

Однако некоторые команды все же могут быть выполнены. К ним относятся команды, при дешифрации которых последний символ С окажется опущенным, например команда

DIR ...,

реализованная в версии 4.0.

Все вместе взятые изменения алгоритма работы системы, описанные выше, вызывают, как правило, у пользователя невысокой квалификации иллюзию поломки компьютера, что приводит к желаемому результату.

Замечание. Следует отметить, что с помощью приемов, представленных в примере 42, защиту можно сделать достаточно сложной. В качестве возможных путей повышения эффективности защиты отметим следующие:

- введение ключей, состоящих из цепочек символов;
- введение комплементарных (дополняющих друг друга) пакетных файлов, позволяющих делать переназначения нестабильными;
- переназначение нескольких клавиш;
- введение имплицативных (условных) переназначений, например в зависимости от дня недели;
- использование клавиш редактирования командной строки.

При желании этот список может быть продолжен каждым пользователем самостоятельно.

3.9. Команда SELECT (инициализация ОС)

ОБЩИЕ СВЕДЕНИЯ. Команда SELECT, реализованная программой, загрузочный модуль которой находится в файле SELECT.COM, появилась в системах MS-DOS и PC-DOS начиная с версии 3.0. С версией 4.0 вместо файла SELECT.COM поставляется набор файлов (SELECT.EXE, SELECT.DAT, SELECT.HLP и SELECT.PRT), реализующих диалоговую процедуру инициализации системы.

SELECT - комплексная процедура настройки операционной среды (хотя далеко не по полному набору параметров). В версиях DOS 3.x SELECT - это минимальная процедура, в процессе которой выполняются следующие операции:

- форматирование целевого диска;
- перезапись ядра и утилит ОС;
- формирование файлов конфигурирования и автонастройки системы - CONFIG.SYS и AUTOEXEC.BAT;
- настройка на специфику конкретной страны.

В системах PC-DOS и MS-DOS 4.0 процедура SELECT предоставляет возможность управлять составом резидента будущей системы, что делает ее отдаленно похожей на процедуру генераций системы (разумеется, процедура SELECT - это не процедура генерации, так как не включает операций получения объектных и загрузочных модулей ядра ОС из исходных текстов).

В версии 4.0 программа SELECT поставляется в формате абсолютной загрузки на отдельной дискете и загружается аппаратным загрузчиком. Эта программа не требует командной строки, а сама опрашивает пользователя в режиме программированных ответов, что делает интерфейс процедуры SELECT с оператором крайне простым.

НАЗНАЧЕНИЕ. Команда SELECT предназначена для инициализации ОС на магнитном диске с учетом специфической настройки на конкретную страну.

ОПИСАНИЕ. Непосредственно после загрузки SELECT выдает предупреждающее сообщение:

SELECT is used to install DOS the first time. SELECT erases everything on the specified target and then installs DOS.
Do you want to continue (Y/N)? Y
(SELECT используется для первоначальной инициализации DOS. SELECT удаляет все содержимое со специфицированного целевого диска, а затем устанавливает DOS. Хотите продолжить (Д/Н)?).

При поступлении ответа "Нет" производится выход из программы. При поступлении ответа "Да" осуществляется контроль параметров и, если параметры заданы корректно, начинается процесс выполнения команды. Если командная строка содержит ошибку, то SELECT выводит одно из следующих сообщений, в зависимости от характера ошибки:

Incorrect number of parameters
(Неверное число параметров);

Invalid parameter

(Неверный параметр);

Invalid drive specification

(Неверная спецификация привода);

Invalid country code

(Неверный код страны);

Invalid keyboard code

(Неверный код клавиатуры).

Далее программа SELECT пытается выполнить команду FORMAT с ключом /S для целевого привода. Для этого SELECT ищет в корневом каталоге исходного привода файл FORMAT.COM. Альтернативные маршруты не просматриваются. Если файл программы FORMAT найти не удастся, то на дисплей выводится сообщение об ошибке

Cannot execute C:FORMAT.COM

(Невозможно выполнить C:FORMAT.COM).

После этого выполнение программы завершается.

При наличии файла FORMAT.COM для целевого привода выполняется команда FORMAT с ключом /S - форматирование и перезапись скрытых системных файлов.

После отрицательного ответа на последний запрос, выдаваемый программой форматирования FORMAT.COM:

Format another (Y/N)?

(Форматировать следующую?),

программа FORMAT завершается и управление возвращается программе SELECT.

Последняя операция процедуры SELECT - перезапись утилит и загружаемых драйверов. Для выполнения этой операции SELECT использует системную утилиту XCOPY. Если файл XCOPY.EXE отсутствует в корневом каталоге исходного диска, то SELECT выдает сообщение об ошибке

Cannot execute x:XCOPY.EXE

(Невозможно выполнить x:XCOPY.EXE),

где x - имя привода магнитных дисков.

В противном случае все файлы из корневого каталога исходного диска переписываются на целевой диск.

ФОРМАТ.

[d:*маршрут*]**SELECT** [[d1:] [d2:][*маршрут*]][ууу][хх]

Спецификация d:*маршрут* задает каталог, содержащий файл загрузочного модуля программы ХСОРУ. Спецификация необходима, если этот каталог не текущий и не альтернативный.

Параметр d1: задает привод, на котором установлен носитель с исходной системой (привод - источник).

Параметр d2: задает целевой привод, на котором система должна быть проинициализирована.

Параметр *маршрут* задает указатель каталога, в который будет переписана ОС. По умолчанию устанавливается корневой каталог.

Параметр ууу задает код страны, который будет установлен для новой системы.

Параметр хх задает код клавиатуры, который будет установлен для новой системы.

Замечания.

1. В качестве исходного привода должны использоваться только приводы гибких магнитных дисков А и В. В противном случае **SELECT** выводит сообщение о некорректности имени привода

Invalid drive specification
(Неверная спецификация привода).

Контроль имени исходного привода введен с учетом того, что команда **SELECT** предназначена для первой перезаписи системы с гибких магнитных дисков. Однако этот контроль можно легко обойти, переназначив магнитный диск с помощью команд **ASSIGN** или **SUBST**. Например, чтобы выполнить процедуру **SELECT** с использованием диска С: в качестве исходного, достаточно предварительно задать одну из двух команд:

ASSIGN B C или **SUBST B C:**

2. Если команда **SELECT** используется для привода высокой емкости (1,2 Мбайт), то должен быть применен соответствующий гибкий магнитный диск.

3. Команда **SELECT** выполняет форматирование с помощью команды **FORMAT**, поэтому при инициализации ОС на целевой дискете вся информация обнуляется.

4. В версиях **DOS 4.0** команда **SELECT** реализована в виде вспомогательной сервисной программы, имеющей свою систему меню. Приведенный выше синтаксис командной строки соответствует версиям **DOS 3.3**.

5. Интерфейсы команды **SELECT** с оператором, реализованные в системах **PC-DOS 3.3** и **MS-DOS 3.3**, полностью совпадают. В версиях **MS-DOS 4.0** и **PC-DOS 4.0** существуют некоторые различия, которые имеют, однако, непринципиальное значение, и поэтому их описание не приводится.

П р и м е р 43.

```
C:\>assign b = c
C:\>select b: a: 61, US
```

SELECT is used to install **DOS** the first time. **SELECT** erases everything on the specified target and then installs **DOS**. Do you want to continue (Y/N)? Y

Insert new diskette for drive A: and strike **ENTER** when ready

Head: 0 Cylinder: 0

Head: 1 Cylinder: 0

Head: 0 Cylinder: 1

Head: 1 Cylinder: 1

⋮

Head: 0 Cylinder: 78

Head: 1 Cylinder: 78

Head: 0 Cylinder: 79

Head: 1 Cylinder: 79

Format complete

System transferred

1213952 bytes total disk space

78336 bytes used by system

1135616 bytes available on disk

Format another (Y/N)?n

Insert **DOS** diskette in drive B:

Strike any key when ready

Reading source file(s)...

B:4201.CPI

B:5202.CPI

B:ANSI.SYS

B:COMMAND.COM

B:EDLIN.COM

B:CONFIG.SYS

B:SCAN.EXE

B:ALFA.EXE

B:COUNTRY.SYS

B:ALFA.SET

B:PR-1012.COM

B:DISPLAY.SYS

B:DRIVERSYS

B:PRE.COM

B:PARAM.LEX

B:LEXE

B:P.EXE

B:EGA.CPI

B:KEYBOARD.SYS

B:LCD.CPI

B:PRINTER.SYS

Reading source file(s)...

B:RAMDRIVE.SYS

B:HDP-910.COM

B:README.DOC

B:FNTEPS.DLF

B:S.EXE

B:SELECT.COM

B:FORMAT.COM

B:XCOPY.EXE

B:AUTOEXEC.BAT

42 File(s) copied

C:\>

В примере 43 показано использование команды SELECT для формирования резервной копии корневого каталога системного диска. В качестве исходного используется жесткий диск C:, предварительно переназначенный на привод B:. Так как корневой каталог исходного диска не содержит большей части утилит ОС, то в результате выполнения процедуры SELECT они переписаны не были. В то же время на целевой диск были переписаны файлы, не имеющие отношения к ОС.

диск были переписаны файлы, не имеющие отношения к ОС. Таким образом, программа SELECT не анализирует перезаписываемую информацию, а только формально запускает команды FORMAT и XCOPY.

В процессе выполнения процедуры SELECT, приведенной в примере, был сформирован файл конфигурирования и автонастройки системы - CONFIG.SYS, состоящий из единственной команды COUNTRY:

COUNTRY = 061,437

и AUTOEXEC.BAT, имеющий следующий вид:

```
PATH \;  
KEYB US 437  
ECHO OFF  
CLS  
DATE  
TIME  
VFP
```

3.10. Команда SET (установка системных определений)

ОБЩИЕ СВЕДЕНИЯ. Команда SET реализует интерфейс оператора с системной областью связи (Environment-окружением DOS).

Системная область связи - это буфер оперативной памяти, предназначенный для накопления строк символьной информации, каждая из которых оформляется в виде отдельной записи. Его адрес система заносит в префикс программного сегмента (PSP - Program Segment Prefix) при загрузке каждой программы в оперативную память. Этот параметр имеет смещение 2CH относительно начала PSP.

Таким образом, наличие окружения DOS предоставляет пользователю удобную возможность запуска программ с различными начальными условиями.

Не трудно видеть, что системная область связи может использоваться в качестве общей области для обмена информацией между программами, каждая из которых оформлена в виде отдельного загрузочного модуля. Максимально возможный размер этой области - 32 Кбайт.

С системной областью связи работают многие утилиты ОС и прикладные программы. Например, распространенной процедурой является получение альтернативных маршрутов с помощью переменных PATH и APPEND.

Некоторые из программ, реализующих выполнение команд ОС, устанавливают в окружении свои переменные. К ним относятся следующие:

SHELL /P - переменная COMSPEC (спецификация файла интерпретатора командной строки);

COMMAND /P - переменная COMSPEC (спецификация файла интерпретатора командной строки);

PATH - переменная PATH (список альтернативных маршрутов, установленных по последней команде PATH);

APPEND /E - переменная APPEND (список альтернативных маршрутов, установленных по последней команде APPEND);

PROMPT - переменная PROMPT (текущий формат подсказки ОС).

НАЗНАЧЕНИЕ. Команда SET предназначена для установки системных определений в специальной области оперативной памяти резидента ОС - окружении DOS. Если в команде SET указано имя параметра со знаком равенства, то строка, задающая соответствующий параметр в окружении, уничтожается. Команда SET, заданная без параметров, выводит значения всех переменных окружения (системной области связи).

ОПИСАНИЕ. По команде SET в системную область связи заносится новая строка символов. Предварительно производится синтаксический контроль этой строки.

Если размер системной области связи не позволяет установить значение переменной, специфицированной в команде SET, то система выводит сообщение об ошибке:

Out of environment space
(Не хватает пространства окружения).

В этом случае необходимо увеличить размер окружения DOS с помощью команды COMMAND или SHELL.

ФОРМАТЫ.

Ф о р м а т 1

Устанавливает значение переменной в системной области связи.

SET имя = значение

Ф о р м а т 2

Убирает определение переменной с указанным именем из системной области связи.

SET имя =

Ф о р м а т 3

Выводит имена и значения всех переменных, установленных в системной области связи.

SET

Параметр *имя* - это произвольная последовательность символов, обозначающая идентификатор параметра, заданного командой SET.

Параметр *имя* не должен содержать знаков равенства (=), а также символов переадресации ввода-вывода (>, <) и конвейерной обработки (|).

Параметр *значение* - произвольная последовательность символов, которая задает устанавливаемую величину параметра или новое назначение.

Параметр *значение* не должен содержать знаков равенства (=), а также символов переадресации ввода-вывода (>, <) и конвейерной обработки (|).

Замечания.

1. Для переменных окружения ОС по умолчанию имеется только 128 байт оперативной памяти. Этот диапазон, однако, может быть расширен до 32768 байт с помощью ключа /E:XXXXX команды SHELL, устанавливающей интерпретатор командной строки COMMAND.COM, или ключа /E:XXXXX команды COMMAND.

2. Между системной областью связи и пакетными файлами существует удобный интерфейс, позволяющий реализовать режим замены переменных их значениями из системной области связи.

Для реализации режима замены в тексте пакетного файла идентификатор соответствующей команды должен быть помещен между знаками процента (%). Например, если одна из строк окружения содержит определение

QWERTY = 102 ,

то запись %QWERTY% повлечет замену имени переменной QWERTY на ее значение из системной области связи - на 102.

3. Если для переменной, заданной в команде SET, было ранее установлено какое-нибудь значение в системной области связи, то оно заменяется новым.

4. Команда SET внутренняя.

П р и м е р 44.

```
C:\>set  
COMSPEC = C:\COMMAND.COM  
PATH = C:\;C:\UDOS;C:\H\TECHRUS;C:\H\HLPCOM;  
PROMPT = $p$g  
C:\>set polz = L132  
C:\>set COMSPEC = C:\COMMAND.COM  
PATH = C:\;C:\UDOS;C:\H\TECHRUS;C:\H\HLPCOM;  
PROMPT = $p$g  
POLZ = L132  
C:\>
```

Первая команда SET, заданная без параметров, вывела текущие значения системных переменных окружения:

- спецификацию файла интерпретатора командной строки (переменная COMSPEC);
- список альтернативных маршрутов (переменная PATH);
- описатель формата системной подсказки (переменная PROMPT).

С помощью второй команды SET устанавливается пользовательская переменная POLZ, что подтверждается справкой, полученной третьей командой SET.

3.11. Команда VERIFY (установка/опрос программного переключателя режима верификации)

ОБЩИЕ СВЕДЕНИЯ. Команда VERIFY позволяет управлять режимом записи информации на магнитные диски с помощью программного переключателя VERIFY. Установкой этого переключателя в состояние ON (включено) запускается контрольная операция считывания после каждой операции записи информации на диск. В этом режиме при обнаружении ошибок на экран консоли выводятся соответствующие сообщения.

В режиме VERIFY ON существенно повышается достоверность выполнения операций записи данных на магнитные диски (понижается вероятность ошибки). При этом, разумеется,

снижается скорость обмена с магнитными дисками при выполнении операций записи информации.

Значение переключателя VERIFY устанавливается, как правило, в зависимости от трех факторов:

- надежности имеющихся устройств внешней памяти на магнитных дисках и компьютера в целом;
- степени ценности информации, записываемой на магнитный диск;
- наличия резервных копий этой информации.

Многие пользователи формируют по две копии наработанной информации, что вполне оправданно. Режим верификации может включаться, например, при получении первой копии и выключаться при получении второй копии.

Следует отметить, что правильная установка (соответственно конкретным условиям) переключателя VERIFY в одних случаях позволяет сохранить данные, а в других - повысить скорость обмена с магнитными дисками. При сомнениях всегда отдавайте предпочтение первому из этих двух факторов.

НАЗНАЧЕНИЕ. По команде VERIFY включается или выключается глобальный режим верификации при выполнении операций записи информации на магнитный диск. Команда без параметров позволяет получить текущее состояние.

ОПИСАНИЕ. Команда VERIFY реализована с помощью двух программ-функций из резидентной библиотеки ОС:

- функции 54H прерывания INT 21H - получения значения программного переключателя верификации операций записи на магнитный диск VERIFY;
- функции 2EH прерывания INT 21H - установки значения программного переключателя верификации операций записи на магнитный диск VERIFY.

Первая функция возвращает текущее значение программного переключателя VERIFY в регистре AL (0 - OFF, 1 - ON), а вторая устанавливает заданное значение этого переключателя соответственно коду, содержащемуся в регистре AL (0 - OFF, 1 - ON).

ФОРМАТЫ.

Ф о р м а т 1

Включение режима верификации при записи информации на магнитный диск.

VERIFY ON

Ф о р м а т 2

Выключение режима верификации при записи информации на магнитный диск.

VERIFY OFF

Ф о р м а т 3

Получение текущего значения программного переключателя режима верификации.

VERIFY

Параметр ON устанавливает режим верификации операций записи информации на магнитные диски.

Параметр OFF отключает режим верификации операций записи информации на магнитные диски. Значение OFF устанавливается по умолчанию.

Если в команде специфицирован параметр, отличный от ON или OFF, то VERIFY выводит сообщение об ошибке

Must specify ON or OFF

(Нужно специфицировать ON или OFF).

Замечания.

1. Команда VERIFY не работает с приводами магнитных дисков, подключенными к внешним компьютерам, функционирующим в сети.

2. Режим контроля записи на диск, аналогичный режиму VERIFY ON (при выключенном режиме верификации), можно установить на время выполнения операций копирования данных с помощью следующих команд:

- COPY с ключом /V;
- XCOPY с ключом /V.

Ключ /V позволяет более гибко управлять состоянием переключателя VERIFY при копировании файлов.

3. Команда VERIFY внутренняя.

Пример 45.

```
C:\> verify  
  
VERIFY is off  
  
C:\> verify on  
  
C:\> verify  
  
VERIFY is on  
  
C:\>
```

После задания первой команды **VERIFY** без параметров система вывела текущее значение программного переключателя режима верификации - **OFF**. Вторая команда **VERIFY** установила значение этого переключателя в положение **ON**, а третья команда показала измененное значение этого переключателя.

3.12. Команда MEM (получение справки о распределении оперативной памяти)

ОБЩИЕ СВЕДЕНИЯ. Команда **MEM** реализована только в версии 4.0 систем **PC-DOS** и **MS-DOS**.

Команда **MEM** позволяет получить информацию о распределении оперативной памяти в системе. Эта команда очень удобна при выполнении системных процедур. Поэтому чаще всего команда **MEM** используется при настройке системы, при включении в систему драйверов внешних устройств, при отладке пользовательских программ и т. п. Команда **MEM** полезна также при возникновении сбойных ситуаций, связанных с нарушениями в работе резидентной части ОС.

НАЗНАЧЕНИЕ. Команда **MEM** предназначена для получения справочной информации об имеющейся в системе оперативной памяти (стандартной, расширенной и внешней), а также обо всех активных и загруженных на текущий момент утилитах, драйверах и прикладных программах.

ОПИСАНИЕ. Для каждой программы (или области данных), размещенной в оперативной памяти, команда **MEM** отображает следующую информацию:

- начальный адрес;
- имя;
- размер;
- тип, характеризующий назначение области памяти.

В заключительной части справки выдается общая информация об имеющейся в системе оперативной памяти:

- общий объем основной памяти;
- общий объем основной памяти, доступной для использования;
- общий объем памяти EMS;
- объем свободной памяти EMS;
- общий объем внешней оперативной памяти.

ФОРМАТ.

[d:маршрут]MEM [/PROGRAM][/DEBUG]

Спецификация d:маршрут задает каталог, содержащий файл загрузочного модуля программы XCOPY. Она необходима, если этот каталог не текущий и не альтернативный.

Ключ /PROGRAM позволяет помимо укрупненной информации о распределении оперативной памяти, выводимой при выполнении команды MEM без ключей, получить информацию обо всех программах и загружаемых драйверах внешних устройств, расположенных в оперативной памяти. При этом выводится информация о распределении памяти для резидентных программ и загружаемых драйверов.

Ключ /DEBUG дополняет справку, выводимую по ключу /PROGRAM, информацией об устройствах, поддержка которых осуществляется резидентными драйверами. При этом выводятся физические имена всех устройств, установленных в системе (CON, AUX, PRN, CLOCK\$, A: - Z:, COM1, LPT1, LPT2, LPT3, COM2, COM3, COM4 и др.).

Замечания.

1. Расширенная память отображается, если система содержит более 1 Мбайт оперативной памяти и установлен драйвер, осуществляющий поддержку стандарта LIM EMS 4.0, например XMA2EMS.SYS.

2. Внешняя память отображается, если объем оперативной памяти, установленной в системе, составляет 1 Мбайт или более.

3. Ключ /PROGRAM и ключ /DEBUG не могут быть специфицированы в одной командной строке. Если эти ключи будут заданы совместно, система выведет сообщение об ошибке.

Too many parameters - /xxxxxxx
(Слишком много параметров - /xxxxxxx),
где /xxxxxxx - имя ключа, введенного вторым.

4. Система не воспринимает сокращенные наименования ключей /PROGRAM и /DEBUG.

5. Команда внешняя, выполняется программой, загрузочный модуль которой содержится в файле MEM.EXE.

Пример 46.

A:\>mem /debug

Address	Name	Size	Type
000000		000400	Interrupt Vector
000400		000100	ROM Communication Area
000500		000200	DOS Communication Area
000700	IBMBIO	002540	System Program
	CON		System Device Driver
	AUX		System Device Driver
	PRN		System Device Driver
	CLOCK\$		System Device Driver
	A: - E:		System Device Driver
	COM1		System Device Driver
	LPT1		System Device Driver
	LPT2		System Device Driver
	LPT3		System Device Driver
	COM2		System Device Driver
	COM3		System Device Driver
	COM4		System Device Driver
002C40	IBMDOS	0088D0	System Program
00B510	IBMBIO	004BE0	System Data
	ANSI	001190	DEVICE =
		0000C0	FILES =
		000100	FCBS =
		0029A0	BUFFERS =
		0001C0	LASTDRIVE =
		000CD0	STACKS =
010100	COMMAND	001A20	Program
011B30	MEM	000030	Environment
011B70	COMMAND	000100	Environment
011C80	MEM	012F60	Program
024BF0	IBMDOS	07B400	-- Free --

655360 bytes total memory
655360 bytes available
582512 largest executable program size

393216 bytes total extended memory
393216 bytes available extended memory

A:\>

4. ПРИМЕРЫ ФАЙЛОВ НАСТРОЙКИ ОПЕРАЦИОННОЙ СРЕДЫ

Ниже представлены примеры файлов CONFIG.SYS и AUTOEXEC.BAT, реализованные в системе MS-DOS 4.01.

4.1. Файл CONFIG.SYS

Пример 47.

```
rem
rem установка количества зарезервированных
rem имен приводов магнитных дисков
rem
lastdrive = q
rem
rem установка количества зарезервированных
rem логических номеров устройств
rem
files = 40
rem
rem установка размера пула буферов
rem
buffers = 20
rem
rem инициализация расширенного драйвера клавиатуры
rem
device = ansi.sys
rem
rem инициализация драйвера переключения таблиц кодов
rem для работы с адаптером EGA
rem
device = display.sys con = (ega)
rem
rem установка расширенного режима прерываний по CTRL-BREAK
rem
break on
rem
```

```
rem расширение системной области связи
rem
shell = c:\command.com /e:256/p
rem
rem установка режима кэширования каталогов
rem
install fastopen.exe c: = 20
rem
rem подключение APPEND к окружению DOS
rem
install append.exe /e
```

4.2. Файл AUTOEXEC.BAT

Пример 48.

```
rem
rem отключение режима эхо-печати команд пакетного
rem файла на экране системной консоли
rem
@echo off
rem
rem вывод номера версии ОС
ver
rem установка альтернативных маршрутов поиска файлов
rem загрузочных модулей EXE, COM, BIN, BAT
rem
path c:\;c:\d;c:\h\techrus;c:\h\hlpcom;c:\norton
rem
rem установка альтернативных маршрутов поиска файлов
rem данных и оверлеев
rem
append c:\h\techrus /E
rem
rem установка системной подсказки и режима отображения
rem текущей даты и времени
rem
prompt $e[s$e[1;60f$e[37;42;1m $t$h$h$h$h$h$h$h $d$e[u$e[35m $e[44m
$p$p$q$g
rem
rem инициализация драйвера для поддержки кириллицы при работе
rem с адаптером EGA
rem
alfa ergha
rem
rem инициализация драйвера печатающего устройства
rem
hdp-910
rem
rem обнуление экрана
cls
```

```
rem автоматическая установка текущего каталога для работы
rem
cd pcc\i2\k4_2
rem
rem резервная конструкция для последующего временного
rem шунтирования некоторых команд
rem
goto 1
:1
rem
rem обнуление экрана
cls
rem установка защиты от несанкционированного доступа к системе
rem после загрузки
rem
prompt $e[0;68;"shilo";13p$e[13;"C";13p$e[s$e[1;60H$e[37;42;1m
$t$h$h$h$h$h$h $d$e[u$e[35m$e[44m$p$q$g
rem
rem установка минимально возможного времени реакции системы
rem при непрерывном нажатии клавиши на клавиатуре
rem
mode con rate = 32 delay = 1
rem
rem выполнение программы фиксации даты и времени
rem на магнитном диске
log
```

Более детальное руководство по ОС DOS и архитектуре ПЭВМ, разработанное автором данной статьи, включает в себя 18 книг и может быть заказано по адресу редакции.

Л.Д. Ганелин

PARADOX3 - ЛИДЕР, НО НЕ В СССР

Существующие СУБД созданы для двух категорий пользователей: профессионалов (программистов) и непрофессионалов (бухгалтеров, инженеров, ...).

1. Paradox3 - для программистов-профессионалов

Профессиональный программист в СССР сейчас может выбрать из нескольких СУБД: FoxPRO, R:BASE, dBASE IV, Paradox, ORACLE, DataEasy, Мастер, ИРИС. Реально на наш рынок персональных СУБД претендентов еще меньше. Например, ORACLE - совершенно неперсональная система, так как один программист не в силах поддерживать ее. (Это касается и других СУБД, перенесенных с больших ЭВМ. Эти СУБД все могут, но от этого не легче.) DataEasy, по слухам, - очень хорошая база, но она потеряна для советского рынка из-за "слишком хорошей" защиты. Фирма перестаралась, удерживая под жестким контролем несколько легальных копий, в то время как остальные СУБД рассеяли семена для будущих всходов.

Не буду сравнивать СУБД. Постараюсь показать хорошие свойства Paradox3.

Для программиста, создающего приложения для персонала, один из самых важных критериев - скорость создания приложений. На Paradox3 в 3-5 раз быстрее, чем на Clipper'e. Почему? Потому что Paradox3 - более объектно-ориентированный инструмент, чем Clipper. Создатели Paradox3 руководствовались правилом: "Все стандартные операции с базами данных должны быть автоматизированы!"

Приведу список операций, автоматизированных Paradox3, облегчающих или целиком заменяющих программирование.

В языке PAL (Paradox Application Language) есть операторы для создания форм и отчетов, как в Clipper'e. Однако используются они крайне редко, так как в Paradox3 есть мощные генераторы форм и отчетов. В программах надо только

указать, какую форму или отчет следует использовать. Таким образом, страницы неудобоваримого описания форм и отчетов заменяются на один оператор. Изменение формы или отчета практически никогда не влияет на текст программы.

Генератор отчетов может

- создать отчеты без программирования;
- разбить отчет по ширине на несколько страниц;
- перенести длинные поля на несколько строк;
- подавить печать повторяющихся значений в нескольких записях;
- "схлопывать" текст при отсутствии значений в полях;
- создать отчет по нескольким таблицам. Для сбора информации из нескольких таблиц не надо делать с помощью запросов временную таблицу, в одном отчете можно использовать данные из нескольких таблиц;
- создать шапки и подвалы отчета, страниц, групп;
- выполнить другие стандартные для генераторов отчетов функции...

Генератор форм может создать

- многотабличные формы. С помощью одной формы удастся просматривать связанную информацию сразу из нескольких таблиц. При просмотре таблицы через такую форму автоматически загружаются связанные таблицы и между ними устанавливаются связи;
- многозаписные формы. Они могут включать в себя многозаписные области для связанных таблиц;
- многоэкранные формы;
- управляемый цвет;
- вычисляемые поля, поля для просмотра и т.д.

Управление видом таблицы на экране (в Clipper'е это - режим **BROWSE**): количество строк на экране, порядок и ширина колонок, формат полей и т.д. Настройка вида таблицы на экране может сохраняться в файле, после чего при загрузке таблицы на экран она примет вид в соответствии с настройкой. Из программы управлять видом таблицы нет необходимости.

Контроль правильности ввода: min и max значения, значение по умолчанию, обязательный ввод значения, мощные шаблоны ввода. Легко задать связь с полем таблицы-справочника для проверки и помощи при вводе. Если задана только проверка, то после ввода в поле значения, оно будет проверено

на наличие в одноименном поле таблицы-справочника. Если задана дополнительная помощь, то, находясь в этом поле и нажав F1, вы попадете в таблицу-справочник, из которой выберете и перенесете в исходную таблицу нужное значение.

Все проверки вводимой информации сохраняются в файле. Практически никогда не надо контролировать вводимые данные с помощью программы.

Ввод данных. Paradox3 поддерживает естественное для индексированных таблиц требование уникальности первичного ключа, т.е. в таблице не может быть более одной записи с одинаковым значением ключевых полей. Поэтому, вводя новые записи, надо контролировать, чтобы они не конфликтовали с существующими записями. В Paradox3 есть унифицированная процедура ввода. Пользователь заполняет пустую таблицу ввода, имеющую структуру, как у дополняемой таблицы, а после этого из таблицы ввода записи начинают передаваться в дополняемую таблицу. При конфликте по ключу записи не вводятся в нее, а поступают в специальную таблицу, которую после ввода можно проанализировать, скорректировать и опять сложить с дополняемой таблицей.

Сложение и вычитание таблиц. При сложении, как и при вводе, первичный ключ проверяется на уникальность. Сложение и вычитание таблиц выполняются с помощью одного оператора.

Ввод сразу в несколько таблиц (так называемые “МногоВвод” и “МногоПлюс”). Таблица ввода содержит поля, соответствующие заполняемым полям в дополняемых таблицах. Если в дополняемых таблицах есть одинаковые поля, то в таблице ввода будет только одно соответствующее поле. Значение же этого поля при вводе будет размножено в соответствующие поля дополняемых таблиц.

Создание и использование макросов. Требуемые действия программы можно смоделировать вручную (с клавиатуры) и внести их в “дежурную” программу (аналог макроса в других системах), далее вставить текст этой программы в нужное место вашей программы.

Использование запросов. Paradox3 обладает мощнейшим языком запросов по образцу (QBE). Можно создать и отладить желаемый запрос, после чего сохранить его в виде текстового файла и вставить в текст вашей программы.

Редактор Paradox3 способен быстро собирать текст программы из заготовленных “дежурных” программ и программ “запросов”.

Кросстабуляция и деловая графика. В сложных приложениях иногда необходим анализ данных. Например: есть таблица с колонками “Станок”, “Изделие”, “Раб.время”. Надо построить таблицу с колонками “Станки”, “Изд.1”, “Изд.2”, ..., в которой, как в футбольной таблице, на пересечении строки “Станок ДИП-500” и колонки “Изд.2” было бы соответствующее рабочее время. Сложность в том, что количество колонок надо определять динамически, во время выполнения программы. В Paradox3 такая задача решается с помощью одной команды “Кросстабуляция”. Так же, с помощью одной команды, по синтезированной таблице будет построена диаграмма.

Связь таблиц. В Paradox3 многие задачи решаются оригинальными, изящными методами. Например, связь таблиц. Paradox3 автоматически устанавливает динамическую связь между таблицами в следующих случаях: при вводе с помощью таблиц-справочников, в многотабличных формах и отчетах, при вводе одновременно в несколько таблиц. Создание связи в Paradox3 - это не самоцель, а средство решения вполне конкретных задач.

Библиотеки, динамическое управление памятью. Paradox3 предоставляет программисту мощные средства создания больших приложений. В PAL есть операторы работы с библиотеками: CreateLib, ReadLib, WriteLib, InfoLib, Release Procs. Можно использовать столько процедур, сколько позволяет оперативная память. Управление памятью в Paradox3 - динамическое. При недостатке памяти неиспользуемые процедуры выгружаются. Paradox3 автоматически использует дополнительную память (> 1 Мбайт), резко увеличивая скорость работы

Управление буферами. Программирующий в Paradox3 избавлен от управления областями. В Paradox3 просто нет такого понятия. При загрузке таблицы в рабочую область для просмотра или редактирования соответствующие буферы выделяются автоматически.

Особенности PAL'а. PAL - традиционный язык, подобный Паскалю, ориентированный на процедурное, структур-

ное программирование. Отметим ряд операторов, облегчающих жизнь программисту баз данных:

- операторы Scan ... EndScan организуют цикл по всем записям таблицы;

- CopyToArray и CopyFromArray используются для хранения записи в массиве. Причем здесь массив - не набор однотипных элементов. Типы элементов определяются типами полей записи, т.е. могут быть разными. Массив нужной размерности и с нужными типами элементов создается автоматически при использовании CopyToArray;

- ShowMenu, ShowFile, ShowTable применимы для создания типичных для Paradox3 однострочных меню.

Отладчик. В Paradox3 есть символьный отладчик для отладки в режиме интерпретации. Это резко ускоряет и упрощает локализацию ошибок. Остановившись на ошибке, практически всегда можно с клавиатуры привести рабочую область в нужный вид и продолжить выполнение программы.

Поддержка изменений. При создании сложной базы данных часто приходится изменять структуры таблиц. Paradox3 максимально облегчает этот процесс. Он автоматически корректирует связанные с измененной таблицей формы и отчеты, а удаленные или измененные поля удаляет или изменяет в соответствующих формах и отчетах, файле вида таблицы на экране и файле проверок вводимых значений.

Оптимизация запросов. Выполняя запросы, Paradox3 оптимизирует их. Например, скорость выполнения запроса по двум таблицам (одна в 50, другая в 50000 записей) в Clipper'e будет сильно зависеть от последовательности этих таблиц, а Paradox3 выполнит запрос в нужном порядке с максимальной скоростью. Чтобы выполнить некоторые запросы, Paradox3 автоматически строит дополнительные индексы. Если вы скажете, что это - чепуха, то, значит, вы никогда не работали с большими базами.

Защита данных. Paradox3 обладает мощной системой защиты: пароли для защиты таблицы от записи; пароли для шифрации таблиц; ограничения редактирования, просмотра таблиц и отдельных полей, изменения форм и отчетов. Паролей несколько: один системный и несколько пользовательских с разными правами доступа.

Работа в сети. Большинство программистов слышали о Paradox3, как о самой лучшей сетевой СУБД. Это действительно так. Встроенные в Paradox3 сетевые средства по цельности и элегантности не знают себе равных. Любая мелочь продумана, ничего существенного не пропущено, ничего лишнего не вставлено. Какие достоинства? Блокировки записей и таблиц, автоматические блокировки, общие и личные каталоги и таблицы, ведение системного журнала (редкое среди персональных СУБД), способность работать в стандартных сетях (IBM Token Ring/Extended Edition, Novell, 3COM, Banyan, Torgus, AT&T Starlan) и любых других сетях, полностью совместимых с DOS 3.X.

Мощная парольная защита и шифрация в сети тоже необходимы. И все это - на фоне быстрогодействия, к которому никто не может приблизиться.

Выводы. Как ни странно, я не рекомендую всем пользователям FoxPRO, Clipper'a или R:BASE переходить на Paradox3. Это однозначно надо делать в двух случаях: во-первых, если вы еще не начали серьезно работать с другой СУБД; во-вторых, если вы начинаете делать сетевые базы данных. Во всех других случаях затраты на интеграцию ваших баз данных и приложений с Paradox3 плюс непонимание коллег и руководства могут превысить ожидаемый эффект от повышения производительности вашего труда.

2. Paradox3 для непрофессионалов

Если разобраться, то у непрофессионального пользователя довольно мало универсальных программ. Например, текстовые редакторы и электронные таблицы. Большинство непрограммистов никогда добровольно не станет работать с программой, говорящей по-английски. Для бухгалтеров это - непреодолимый барьер, для инженеров - трудный. Непрофессионалы сейчас пользуются чаще всего Лексиконом Веселова и русификатором Framework Чижова.

И все же рискну утверждать, что Paradox3 скоро также станет инструментом непрофессионала. Аргументы такие:

- во-первых, у Paradox3 простой интуитивный пользовательский интерфейс. Ни одна из известных в СССР СУБД аналогичного класса (FoxPRO, R:BASE, dBASE IV, Мастер)

не может сравниться с Paradox3 по этому критерию. В Paradox3 практически отсутствует командный режим. Можно задать русские имена полей, использовать любые русские символы в отчетах и формах. Работа во всех режимах Paradox3 - при просмотре, вводе, создании форм, отчетов, запросов - выполняется по однотипным правилам. Не зря Paradox3 иногда называют интегрированным пакетом наравне с Framework. Paradox3 всегда ведет пользователя "за руку", задавая вопросы до тех пор, пока не поймет, что надо делать.

- во-вторых, Paradox3 не требует от пользователя знания терминологии баз данных. В нем вообще нет понятия область. Можно обойтись без понятия "индекс". Не создавая вторичный индекс, пользователь просто "оптимизирует" (ускоряет) конкретный запрос;

- в-третьих, язык запросов по образцу QBE резко упрощает поиск в базе. Причем, если для других СУБД QBE был включен в поздних версиях (dBASE IV), то для Paradox он - первооснова и не смотрится как приятный, но необязательный довесок, часто усложняющий работу;

- и, наконец, в-четвертых, последнее и решающее преимущество. Только для Paradox3 создан полноценный русификатор. Для других СУБД русификаторов либо вообще нет, либо они сделаны для устаревших версий СУБД. Из собственной практики: после четырехдневных курсов бухгалтеры, никогда не имевшие дела с ПЭВМ, могли самостоятельно создавать таблицы, вводить в них данные и делать по ним простые запросы. После 3-месячной работы с Paradox3 желательно еще неделю учиться профессиональным методам. Это связано с тем, что редкий пользователь в рабочих условиях возьмется за самообучение, даже имея под рукой документацию.

(P.S. По неофициальным данным, фирма Borland планирует начать продажу Paradox3 за рубли.)

Выводы: При резком росте числа непрофессионалов за клавиатурой ПЭВМ число пользователей Paradox3 (с русификатором) станет намного больше. Альтернативы Paradox3 для пользователя-непрофессионала сейчас нет.

ИНТЕГРАЦИЯ PARADOX'a в ПРОГРАММНЫЕ СРЕДЫ

Если Вы еще не используете Paradox для работы со своими данными или приложениями, то спешите... Вы можете отстать в бесконечной гонке за лидером. Пусть вас не смущают вопросы перехода с FoxBASE или с Clipper а в другую среду. Игра стоит свеч. Ваши старые базы и приложения не надо выбрасывать, не надо переписывать, их можно спокойно интегрировать в более современную среду Paradox'a.

Средства интеграции данных и приложений разнообразны и всеобъемлющи.

1. Export/Import. В Paradox встроены фильтры для большинства фактических стандартов форматов баз данных и электронных таблиц.

Фильтры Экспорта/Импорта Paradox'a:

- dBASE II, III и III Plus (файлы .DBF),
- Lotus 1-2-3 версии 1A и 2 (файлы .WK и .WK1),
- Symphony версии 1.0 и 1.1 (файлы .WRK и .WR1),
- Quattro (файлы .WKQ),
- Reflex (файлы .RXD),
- Visicalc (файлы .DIF),
- pfs и IBM Filing Assistant file (файлы .PFS),
- ASCII (файлы .TXT).

2. Paradox Engine - уникальное решение извечной проблемы, открывающее архитектуру Paradox'a для программирования на языке С. Это средства программирования высокого уровня, позволяющие объединять С-приложения с мощностью Paradox'a. Стратегия фирмы Borland - достичь совместимости между их системами для приложений и языками Turbo-семейства. Engine - это ключевое звено в стратегии фирмы.

Выделенные из ядра Paradox'a система управления данными и многопользовательская параллельная технология в виде Paradox Engine дают возможность программировать с помощью более чем 70 функций Интерфейса Программирования Приложений (API - Applications Programming Interface). Эти функции позволяют манипулировать таблицами Paradox'a и в однопользовательской, и в многопользовательской средах.

Engine поставляется в форме С-библиотеки, которая способна связываться с любыми С-программами. Так как Paradox Engine - библиотека, то приложения, написанные с помощью Engine, могут быть очень маленькими. Размер приложений зависит от числа строк кода и количества использованных функций Engine. Так как полученное приложение - обычный выполняемый файл (.EXE), то от него можно ждать преимуществ, характерных для С: в размере и скорости.

Данные - это сердце информационных систем, но, например, большинство средств обработки данных могут работать только с некоторыми форматами данных: файлы из одной среды надо переводить в формат другой среды и наоборот. При попытке пользователей объединить информацию из нескольких систем - новые мучения. Как поддержать интегрированность данных? Как избежать конфликтов при обмене данными?

Engine устраняет эти ограничения, открывая архитектуру данных Paradox'a для С-программ. Ваши С-приложения теперь параллельно с работающими Paradox-приложениями смогут пользоваться существующими данными Paradox'a. Для вашей задачи можете выбрать соответствующее средство, а различные наборы данных не разрастутся.

Paradox Engine использует API, который позволяет вашим С-программам:

- создавать, читать и записывать таблицы, записи и файлы Paradox'a;
- поддерживать ясную блокировку записей и файлов и параллельное выполнение совместно с интерактивным Paradox'ом и PAL-приложениями (написанными на Языке Приложений Paradox'a - Paradox Application Language) так же хорошо, как и с другими Engine-приложениями;
- использовать такие возможности Paradox'a, как защита паролями, шифровка таблиц, кодирование данных, поиск и управление ошибками;
- импортировать данные в таблицы Paradox'a через последовательный порт при загрузке их из главной машины или из внешних устройств, которые не достижимы из PAL'a;
- создавать обычные приложения или приложения, запускаемые PAL-командой RUN;
- иметь неограниченную, свободную от платы лицензию.

Вы теперь можете выбрать, какая часть вашего приложения больше соответствует возможностям С, а какая лучше выполняется с помощью Paradox а или PAL'a.

3. Сети. Paradox и приложения, созданные с помощью Engine, могут работать с большинством сетей, распространенных на персональных компьютерах (IBM Token Ring/Extended Edition, Novell, 3COM, Banyan, Torus, AT&T Starlan), и с любыми другими сетями, полностью совместимыми с DOS 3.X.

4. SQL Link - это интерфейс для трансляции QBE-запросов в SQL-запросы и обратно. С его помощью ваши данные без проблем можно использовать в любой сетевой информационной системе.

5. Clipper driver. Фирма Nantucket объявила о разработке Database Driver (драйверов для пакета Clipper). Драйвер для пакета Paradox и драйверы SQL для разработанного фирмами Ashton-Tate и Microsoft инструмента SQL Server (который может объединять в рамках пакета Clipper запросы к базам данных) будут поставляться после появления пакета Clipper 5.0.

6. Run Time и компилятор. Совершенно необязательно загромождать всю оперативную память компьютера Paradox'ом. Если уже есть готовые PAL-приложения, то можно ограничиться Paradox Run Time - кусочком целого Paradox'a, чтобы "проигрывать" готовые PAL-приложения. Помимо экономии памяти и выигрыша в скорости (как следствие экономии памяти и уменьшения времени на своппинг оверлеев) можно солидно сэкономить деньги, так как Run Time стоит в 20 раз дешевле самого Paradox'a! Run Time позволит использовать PAL-приложения в составе интегрированных информационных систем.

Фирма TRW недавно создала компилятор для языка PAL.

7. Особенности СУБД Paradox'a, облегчающие интеграцию. Существуют версии Paradox3 для MS-DOS и для OS/2, есть даже версия Paradox/386. Paradox сам распознает наличие расширенной памяти и сопроцессора и автоматически их использует. Средства настройки позволяют изменять характеристики Paradox'a в широких пределах (от форматов данных по умолчанию до цветовой палитры). Paradox, как и все продукты фирмы Borland, чрезвычайно неприспособлен к аппаратуре.

1. P3R FULL RUSSIAN - русификатор Paradox3

Функции русификатора Paradox3:

- меню на русском языке с русскими hot keys (одно- временно действуют и прежние английские);
- сообщения на русском языке;
- полная подсказка (help) на русском языке (на 260 экранов);
- словарная сортировка;
- названия месяцев и дни недели только по-русски при вводе-выводе и в функциях PAL (dateval, fieldstr, strval, moy, dow, format);
- векторные шрифты на русском языке для деловой графики (10 шт.);
- менее 40 Кбайт в ОЗУ и менее 15 Кбайт в следующей версии;
- полноценная работа всех программ на языке PAL, написанных для родной версии Paradox;
- нормальная работа всех подсистем Paradox с русскими буквами, включая функции PAL (match, upper, lower, format и т.д.);
- нормальная работа всех шаблонов PAL.

Авторы - коллектив КоКом под управлением Павла Зелинского (при техническом содействии Антона Чинова).

Цена одной копии - 2500 руб.

Распространители: СП ПараГраф, Москва, (095)200-25-66; НПК Дельта, Самара, (8462)661-375.

2. Электронный справочник по Paradox3

Справочник включает в себя полную фирменную документацию по Paradox3, организованную в виде оглавления с раскрывающимися статьями.

Состав справочника по Paradox3:

- введение в систему Paradox;
- представление данных в системе Paradox;
- руководство администратора сети, пользователя по PAL, системе Paradox и системе Personal Programmer.

Авторы - коллектив под руководством Андрея Тищенко.

Цена одной копии, включающей в себя 11 дискет форматом 360 Кбайт, - 3000 руб.

Распространители: МКФ ЭЦ Гермес, Москва, (095)138-26-95; НПК Дельта, Самара, (8462)661-375.

УДК 681.3.06

Ф. Н. Шерстюк

ВИРУСЫ И АНТИВИРУСЫ НА КОМПЬЮТЕРЕ IBM PC

Не будем выяснять, что такое компьютерный вирус: пожалуй, слишком много пользователей персональных ЭВМ (ПЭВМ) о них хорошо знают, а те, кто с ними не знакомы, скоро познакомятся. Вирусы все шире входят в нашу жизнь. Откуда? По-моему, их пишут хулиганствующие программисты, в основном, чтобы потешить свое самолюбие или заработать деньги на продаже антивируса.

Несколько не утопических советов

Вот несколько советов, которые в различных вариациях уже не раз встречались на страницах компьютерных изданий, но которые никогда не мешает повторить.

Лучший способ лечения - не болеть. Пользуйтесь только проверенными программами. Абсолютной гарантии, конечно, никто дать не может, но программа с фирменной дискеты или программа, которой ваш хороший знакомый регулярно пользовался не менее полугода, вероятно, не инфицированы.

Помните: самые распространенные источники вирусов - это игры и антивирусы. Избегайте коллекционировать программы, которыми вы не пользуетесь. Пусть этот совет не покажется вам утопией. Люди, которые ему следуют, есть. Как правило, это люди, использующие компьютер для какой-то реальной работы: инструментарий у них сложившийся (для пополнения нужны очень веские основания), места на диске для коллекционирования ненужных программ у них нет (как и времени для компьютерных игр).

На защищенные от последующей записи дискеты необходимо скопировать все дорогие вам файлы, не только исполняемые (с расширениями EXE, COM, драйверов, оверлеев и т.п.), но и текстовые. Почему? Во-первых, некоторые вирусы тем или иным способом уничтожают файлы всех сортов, а, во-вторых, существуют вирусы, вносящие тонкие, не сразу заметные изменения в текстовые файлы. Особое внимание уделите копи-

ям ОС. Их должно быть не менее двух, и защита с этих дискет никогда не должна сниматься.

Если вирус все же проник на ваш компьютер, то главный совет, который тут можно дать, - не паникуйте, обдумайте дальнейший план действий и только после этого начинайте действовать.

Наиболее надежный и простой способ изгнать вирус (не подцепив при этом нового) таков. Перезагрузитесь кнопкой RESET или выключением-включением питания (а не комбинацией Ctrl+Alt Del) на незараженную копию системной дискеты, отформатируйте все незащищенные диски и дискеты (существуют вирусы, которые цепляются не к файлам, а к дискам и дискетам), а затем восстановите все с защищенных копий. Хотя этот способ и наиболее трудоемкий, пусть и этот совет не покажется вам утопией. Мне известно место, где подобная процедура проводилась на десяти компьютерах приблизительно раз в неделю (так часто там появлялись новые вирусы).

Антивирусы

Скорее всего, данные советы помогут вам не в полной мере и тогда вы, вероятно, захотите прибегнуть к помощи антивирусов. Рассмотрим вкратце достоинства и недостатки специализированных и универсальных антивирусов.

Специализированные - это большинство антивирусных программ: AIDSTEST, VDEATH, SERUM3, ANTI-KOT, SCAN и сотни др. Каждая из них распознает один или несколько конкретных вирусов, никак не реагируя на присутствие остальных. Лишь некоторые из этих антивирусов могут вылечить зараженный файл, но их с упорством, достойным лучшего применения, все создают и создают программисты, специализирующиеся на антивирусах.

Недостаток их: слишком узкая специализация (вирусов уже сейчас очень много и число их стремительно растет, в частности за счет мутаций - изменений, легко вносимых хулиганствующим программистом в вирус, который после этого не опознается соответствующими антивирусами), низкая надежность (такие программы часто пишут в спешке), высо-

кая вирулентность (антивирусы этого типа сами очень часто инфицированы).

Универсальные антивирусы предназначены для борьбы с целыми классами вирусов. Чем отличается универсальный антивирус от специализированного? Специализированный способен бороться только с уже написанными вирусами, а универсальный - и с еще не написанными. Из универсальных мне известны лишь резидентные антивирусы и программы-ревизоры.

Резидентные антивирусы (ANTI4US, FLUSHOT, FPROT, CDM и, может быть, какие-то другие), обнаружив подозрительные действия и запросив у пользователя их подтверждения, пытаются не допустить размножения вирусов.

Недостаток этих программ очевиден: ужасная назойливость. Ведь действия, требующие подтверждения, в среднем не реже чем раз в пять минут совершаются вполне невиновными программами, а вовсе не вирусами. Надежность этих антивирусов невелика, так как авторы вирусов разработали несколько весьма простых и эффективных способов обхода таких программ. Кроме того, эти программы - резидентные и поэтому "отъедают" много оперативной памяти и вступают в клавиатурные конфликты с другими программами.

На ПЭВМ коллективного пользования такие программы практически бесполезны. Почему? В момент заражения за компьютером практически всегда сидит недостаточно квалифицированный пользователь, который со спокойной душой и полным непониманием происходящего дает подтверждение на запрашиваемое действие. После этого бороться с вирусом почти невозможно, так как обычно даже неизвестно, какие программы он успел заразить. Другими словами, эти антивирусы совершенно не обладают "защитой от дурака".

Программы-ревизоры сравнивают параметры файлов (длину, контрольную сумму и т.п.) с эталонными, обнаруживая тем самым, подвергался ли файл несанкционированным изменениям. Однако такие программы работают довольно долго, поэтому ревизии проводят, увы, не чаще одного раза за сеанс работы с компьютером либо не над всеми файлами.

Ни один известный мне универсальный антивирус не способен излечить зараженный файл (а с помощью резидентного антивируса невозможно даже установить, что файл заражен). Из вышесказанного можно (и нужно) сделать печальный вывод, что антивирусные программы пока что либо неэффективны, либо опасны (чаще и то и другое вместе).

Почему современные антивирусы не эффективны и опасны?

На мой взгляд, причины этой трагической ситуации таковы.

1. Борьбу с вирусами ведут в основном программисты невысокой квалификации. Это естественно: у квалифицированных программистов вирусы, как правило, не заводятся (см. первый очевидный совет в начале статьи).

2. В антивирусной борьбе сейчас наиболее популярен “индивидуальный террор” (специализированные антивирусы). Исторический опыт гласит, что этот метод проистекает от неверия в полную победу и неизбежно приводит к полному поражению. Количество и качество универсальных антивирусных программ в настоящее время серьезно уступает количеству и качеству специализированных. Это серьезная методологическая ошибка.

3. Другая методологическая ошибка - непонимание того, что свободу пользователя никакая антивирусная программа не должна сковывать или ограничивать (ведь она предназначена для его удобства!)

4. Парадокс, но все сколь-нибудь серьезные антивирусы могут быть эффективно использованы лишь квалифицированными программистами, хорошо понимающими механизмы размножения вирусов. Но ведь как раз им такие антивирусы и не нужны: вирусы у них почти не заводятся. Хороших антивирусов, ориентированных на “простого пользователя” или хотя бы достаточно надежно “защищенных от дурака”, я пока не видел.

5. Безответственные люди (их компетентность, кстати, обычно невысока), обладающие информацией об устройстве вирусов, широко распространили ее (даже в печати!). Результат - резкий рост числа мутаций импортных вирусов и появление отечественных.

6. Вирусы выгодны тем, кто продает антивирусы (или свои услуги по обеззараживанию компьютеров), пишет статьи о вирусах или хочет свои огрехи списать на вирусы.

Как перехватить инициативу?

Недопустимо упускать инициативу из своих рук и отдавать ее авторам вирусов. Но пока и с той, и с другой стороны квалификация программистов одинаково средняя или низкая, инициатива останется у нападающих, т.е. у тех, кто пишет вирусы. Несмотря на обилие вирусов, в настоящее время идей, по-

ложенных в их основу, совсем немного. Беда в том, что идей, известных мне для конструирования антивирусов, также очень мало (и все они довольно старые). Хочу поделиться несколькими своими соображениями.

Все известные мне вирусы можно разделить на три категории:

специализированные системные вирусы, изменяющие системные файлы (IO.SYS, MSDOS.SYS, IBMI0.COM, IBMDOS.COM, COMMAND.COM), не удлиняя их (наименее распространены);

загрузочные вирусы, изменяющие boot-сектор и перехватывающие int 13 (достаточно широко распространены);

вирусы, удлиняющие EXE и COM файлы (наиболее распространены).

Каждому из вирусов необходим индивидуальный подход, но общая идея для соответствующих антивирусов такова. Эксплуатация резидентных универсальных антивирусов показывает, что мы действительно не можем запретить вирусу размножаться. Если это так, то не надо и пытаться это делать. Но факт заражения надо выявлять на самой ранней стадии. Желательно также уметь удалять вирус с зараженного файла или диска.

Обе эти цели достижимы, если у нас достаточно информации о незараженном состоянии файла или диска и мы часто сравниваем объект заражения с эталоном. Посмотрим, как эту идею можно применить к каждому из перечисленных типов вирусов.

Поскольку системных файлов всего три, нетрудно хранить их копии и запускать программу сравнения копии с оригиналом в AUTOEXEC.BAT.

Загрузочный вирус перехватывает прерывание int 13 еще до начала загрузки DOS. Поэтому для обнаружения такого вируса необходимо знать, на что указывает вектор int 13 в момент начала загрузки (с помощью не очень сложной трассирующей программы этого можно добиться). Если на ПЗУ, то все нормально, а если на ОЗУ, то вектор кем-то перехвачен (вероятно, вирусом). Сохраняя копию boot-сектора и адрес в ПЗУ, на который должен указывать вектор int 13, мы сможем легко излечить компьютер от загрузочного вируса.

Если к каждому EXE и COM файлу в конец дописать стража, которому в момент запуска файла передается управление, то страж проверит, не удлинился ли файл, к которому он прицеплен, т.е. каждый файл, защищенный стражем, в момент запуска сам станет проверять, заражен он или нет. Назовем уязвимым местом область файла, которая скорее всего бу-

дет попорчена вирусом. Если в теле этого стража хранить также контрольную сумму и копии уязвимых мест файла, то, основываясь на этой информации, зараженный файл можно восстанавливать. Это возможно, когда вирус не повредил никаких мест, кроме уязвимых.

Программ сравнения файлов уже немало. Программа для установки стража (защищенный файл удлинится примерно на 1,8 Кбайт) мной написана. Ее успешно эксплуатируют в нескольких организациях. Страж всегда обнаруживает факт заражения и в большинстве случаев позволяет вылечить - восстановить зараженный файл. Что же касается программы для защиты от загрузочных вирусов... Ее надо писать.

Немного демистификации

Чтобы лишить мистического ореола способности вирусов к размножению, приведу некоторые "научные" данные на эту же тему. Сначала договоримся о терминологии.

Компьютер считается зараженным вирусом, если в его оперативной памяти есть резидентная часть этого вируса (т.е. часть вируса, предназначенная для оперативной памяти).

Выполняемый файл (или системный файл, который, в сущности, тоже выполняем) считается зараженным вирусом, если в нем есть часть вируса, предназначенная быть в файле.

Диск (дискета) считается зараженным загрузочным вирусом, если в нем есть часть загрузочного вируса, предназначенная быть на диске.

Вирус не опасен, пока не заражен компьютер (хотя, быть может, заражены находящиеся в нем файлы или вставленные в него дискеты).

Компьютер заражается после того, как на нем запущено что-нибудь зараженное. Практически это означает, что компьютер заражается вирусом, поразившим файл, при запуске этого файла, а загрузочным вирусом - после загрузки с зараженной дискеты.

Все прочие действия (копирование зараженных программ, запуск незараженных программ с зараженной загрузочным вирусом дискеты и т.п.) не опасны для компьютера.

Если компьютер заражен, то, потенциально, все, что с ним соприкасалось, также может заразиться. А именно, если компьютер заражен вирусом, заражающим выполняемые фай-

лы, то зараженным может оказаться любой из этих файлов (даже если этот файл после заражения компьютера ни разу не выполнялся или к нему не было никакого обращения, или он защищен от записи). Если компьютер заражен загрузочным вирусом, то всякая не защищенная от записи дискета, которая в него вставлялась, также может заразиться.

Если компьютер заражен, то после перезагрузки с помощью Ctrl+Alt+Del он также может остаться зараженным. Компьютер перестает быть зараженным только после перезагрузки с незараженной системной дискеты с помощью клавиши Reset (INIT на "Искре 1030") или выключения-включения питания.

Рекомендую все мероприятия по обеззараживанию файлов, пораженных вирусом, начинать именно с такой перезагрузки (иначе, как показывает опыт, иногда можно значительно увеличить количество зараженных файлов). В случае загрузочных вирусов этот совет не всегда полезен, о чем будет рассказано ниже.

Теперь кратко остановимся на тех общих признаках, которые характеризуют различные типы вирусов и которые с помощью простейших подручных средств может легко обнаружить каждый пользователь.

Признаки вирусов, поражающих системные файлы, мне не известны, поскольку вирусы эти чрезвычайно редкие (ни я, ни мои знакомые их не видели).

Каков признак заражения компьютера вирусом, заражающим файлы? Выполняемые файлы увеличиваются в размерах. Это можно обнаружить командой DIR (или с помощью ps или xtree и т.п.), если, конечно, помнить изначальные длины этих файлов (чего я, например, никогда не делаю - за меня это делает страж).

Есть ли признак того, что компьютер заразился загрузочным вирусом? Есть. Объем памяти, доступной DOS, становится меньше, чем был раньше. Этот факт можно обнаружить, например, с помощью программы ptools. Она сообщает, сколько памяти есть на самом деле (т.е. 640, или 512, или 256 Кбайт - эту цифру никакой вирус изменить не в силах) и сколько доступно DOS. Если компьютер не заражен, то для DOS доступна вся память (из этого правила есть редкие исключения, например, у PS/2-50 есть 640 Кбайт памяти, а для DOS доступно 639). Если компьютер заражен, то для DOS доступно памяти меньше, чем ее есть на самом деле. Этот признак есть у всех известных мне загрузочных вирусов, хотя в его абсолютной универсальности я не уверен.

Как пишут антивирусы

В антивирусной борьбе (как и в любой другой деятельности) самое опасное - это монополизм. Ведь против одного, даже самого замечательного, антивируса всегда можно написать специализированный вирус, который его обходит, поэтому антивирусов должно быть много. Далее будет рассмотрена основная техническая информация, необходимая для того, чтобы написать антивирусы по предложенной методике. Эта информация, применяемая в моем методе борьбы с вирусами (хотя, конечно же, создание и других методов весьма желательно), вряд ли поможет при разработке вирусов. Однако, возможно, она уберет их авторов от некоторых ошибок, весьма неприятных для пользователей.

Устройство стража

1. Где размещать стража? Желательно в конце файла. Это относится как к EXE, так и к COM файлам. С чем это связано? Тогда различные настройщики, инсталляторы и т.п., работа которых тесно связана со структурой защищаемого файла, смогут беспрепятственно вносить в него изменения.

Частный случай такой настройки - настройка адресов, которой подвергается всякий EXE файл перед началом работы. Информация, необходимая для этой настройки, находится в relocation table в начале EXE файла и имеет довольно несложный формат. Однако видоизменять ее не рекомендуется, поскольку эта таблица может нести важную информацию для настройщиков.

При размещении стража в конце файла важно помнить, что длина COM файла должна быть меньше 64 Кбайт (65536 байт). Это связано с тем, что в начале работы COM файла длина его стека равна:

<длина стека> = 64 Кбайт - <длина COM файла>.

Поэтому на достаточно длинные COM файлы ставить стража (равно как и вирус) вряд ли имеет смысл.

Прежде чем обсуждать установку стража (и вируса) на EXE файл, рассмотрим его структуру. Всякий EXE файл начинается с заголовка, содержащего служебную информацию, необходимую при его загрузке. Длина заголовка всегда кратна 16. Часть EXE файла непосредственно за заголовком - это и есть выполняемая программа. Содержимое первых 24 байт заголовка изложено в таблице.

Поле SIGN имеет очень важное значение. Именно оно (а вовсе не расширение EXE) отличает EXE файл от COM файла. Это означает, что если файл имеет расширение EXE, но содержимое поля SIGN у него не 'MZ', то он при загрузке будет рассматриваться как COM файл (существует вирус, который этот факт использует).

Отметим такой важный факт (о котором не знают авторы некоторых вирусов). Длина EXE файла вовсе не обязательно совпадает с $512 * (\text{PAGECNT} - 1) + \text{PARTPAG}$ (т.е. с информацией о длине, находящейся в его заголовке). Эта хранящаяся в заголовке информация всего лишь означает, что перед началом выполнения EXE файла в память будет загружено $(512 * (\text{PAGECNT} - 1) + \text{PARTPAG}) - 16 * \text{HDRSIZE}$ байтов. С остатком файла выполняемая программа может поступать по своему усмотрению. Наиболее часты следующие случаи:

- с остатком никак не поступают, так как он появился в результате ошибки (неопасной) редактора связей;
- файл - самораспакованный архив, тогда в остатке содержится собственно архив;
- файл - оверлейная программа, тогда в остатке - оверлеи;
- в остатке находится информация, необходимая для отладчика.

Смещение поля от начала EXE файла	Длина поля, байт	Содержимое поля	Название поля
0 (00h)	2	4Dh 5Ah (т.е. 'MZ') - сигнатура EXE файла	SIGN
2 (02h)	2	Длина недозаполненного сектора (сектор - 512 байт)	PARTPAG
4 (04h)	2	Длина EXE файла в секторах (включая недозаполненный)	PAGECNT
6 (06h)	2	Количество элементов в relocation table (для нас несущественно)	RELOCNT
8 (08h)	2	Длина заголовка в параграфах (параграф - 16 байт)	HDRSIZE
10 (0Ah)	2	Минимальный объем дополнительной памяти, требуемой EXE файлу, в параграфах	MINMEM

12 (0Ch)	2	Максимальный объем дополнительной памяти, требуемой EXE файлу, в параграфах	MAXMEM
14 (0Eh)	2	Начальное значение сегмента стека (SS)	RELOSS
16 (10h)	2	Начальное значение указателя стека (SP)	EXESP
18 (12h)	2	Контрольная сумма (обычно 0, но иногда такое число, что сумма всех слов файла составляет 0)	CHKSUM
20 (14h)	2	Адрес, с которого начинается выполнение	EXEIP
22 (16h)	2	Начальное значение сегмента кода (CS)	RELOCS

Поскольку страж (и вирус) должен находиться в конце EXE файла и перед выполнением загружаться в память, то, кроме записи соответствующего "довеска" в конец файла, необходимо изменить поля PARTPAG и PAGECNT так, чтобы величина $512 * (PAGECNT - 1) + PARTPAG$ равнялась величине удлинённого EXE файла, т.е. положить

$$PARTPAG = \text{MOD}(\langle \text{длина_файла} \rangle + 511, 512)$$

$$PAGECNT = \text{INT}(\langle \text{длина_файла} \rangle / 512)$$

Опыт показывает, что такое изменение заголовка EXE файла (как и все те изменения, которые будут предложены ниже) не сказывается на работоспособности защищаемой (заражаемой) программы. Есть лишь одно исключение. Поскольку длина загружаемой части файла теперь равна длине файла (за вычетом длины заголовка), то он может просто не уместиться в оперативной памяти и поэтому не сможет быть выполнен. Это может случиться с очень большим оверлейным файлом или с очень большим самораспаковывающимся архивом. В таком случае защищать (и инфицировать) файл нецелесообразно.

2. Как страж получает управление. Прежде всего выясним, что происходит при запуске программы.

Управляющая программа (обычно COMMAND.COM, но может быть отладчик, Windows, Make и т.п.) выбирает свободный участок памяти, куда будет загружена программа. Участок начинается с границы параграфа (т.е. с 16-байтной границы). Этот параграф называется PSP (Program Segment Prefix). Первые 256 байт этого участка содержат служебную

информацию, затем загружается сама программа. Далее все происходит в зависимости от типа файла.

Если загружается COM файл, то регистры устанавливаются так:

CS = PSP

DS = PSP

ES = PSP

SS = PSP

AX = <некоторая несущественная для нас, но, может быть, существенная для программы информация>

SP = FFFh (или меньше, если доступно менее 64 Кбайт памяти)

IP = 100h (т.е. выполнение начинается с адреса 256)

При загрузке EXE файла, регистры устанавливаются так:

DS = PSP

ES = PSP

CS = RELOCS+PSP+10h

SS = RELOSS+PSP+10h

AX = <некоторая несущественная для нас, но, может быть, существенная для программы информация>

SP = EXESP

IP = EXEIP

Таким образом, чтобы передать управление стражу, необходимо в случае COM файла записать в самое начало файла JMP на стража, а в случае EXE файла – в поля заголовка RELOCS и EXEIP соответствующие константы. Есть риск, что стек программы будет размещен в том же месте, где находится страж, а это выведет стража из строя. Поэтому желательно также записать соответствующие константы в RELOSS и EXESP.

3. Как страж проверяет файл, который он стережет. Чтобы проверить, например, длину файла, страж должен знать полное имя файла. Рассмотрим способ выяснения имени. Сразу заметим, что это можно сделать не всегда.

В сегменте PSP слово со смещением 44 (2Ch) от его начала содержит сегмент, в котором находится environment. Environment – это набор переменных (в отечественной литературе именуемых “переменными среды”, или “переменными окружения”), переданных выполняемой программе управляющей программой. Вместо того, чтобы давать более “научное” определение, приведу примеры “переменных окружения”: COMSPEC, PATH, PROMPT, CHFILES и т.п. Структуру environment я тоже поясню на примере, который дам в мнемонике ассемблера (не потому, что мне нравится

этот язык, а потому, что в данном случае она наиболее проста):

```
; Начало стандартного environment
db 'COMSPEC = C:\COMMAND.COM',0      ; "Переменная среды"
                                       COMSPEC
db 'PATH = C:\;C:\DOS;C:\BIN',0      ; "Переменная среды"
                                       PATH
db 'PROMPT = $p$g',0                 ; "Переменная среды"
                                       PROMPT
db 'CHIFILES = c:\CW',0               ; "Переменная среды"
                                       CHIFILES
db 0                                  ; Терминатор environment
; В этом месте кончается стандартный environment и начинается не-
; стандартный
dw 1                                  ; Количество элементов
                                       ; в нестандартном
                                       ; environment (обычно 1)
db 'C:\MYDIR\MYPROG.EXE',0           ; Имя программы
```

Как видно, имя программы находится в нестандартной части environment. Не всякая управляющая программа добавляет к environment нестандартную часть. Например, этого не делает COMMAND.COM в версиях DOS, предшествовавших 3.0, этого не делает мой любимый отладчик AFD и, возможно, некоторые другие программы. Чтобы понять, добавлена ли к environment нестандартная часть, необходимо как-то проинтерпретировать слово, идущее за терминатором стандартного environment. Если это слово содержит 0, то вопросов нет (нестандартный environment отсутствует). Если 1, то, вероятно, (хотя и не точно), что нестандартный environment присутствует. А если не 0 и не 1, то непонятно (ведь это может быть нестандартный environment, содержащий более одного элемента).

Если имя узнать не удалось, то страж ничего не предпринимает, а если удалось, - проверяет длину файла.

4. Как страж возвращает управление. Закончив работу, страж должен вернуть управление программе.

В случае COM файла это очень просто - надо восстановить содержимое регистров CS, DS, ES, AX, а также (в памяти, не в файле!) те байты в начале программы, которые были затерты командой JMP (см. п.2. Как страж получает управление), после чего передать управление по адресу PSP:100h.

В случае EXE файла необходимо сначала восстановить содержимое регистров DS, ES, AX, а затем регистров CS, IP,

SS, SP. Здесь есть тонкая проблема. Ясно, что на восстановление регистров SS и SP (переключение стека) необходимы как минимум две команды и как минимум одна команда (FAR JMP), чтобы восстановить CS и IP. Но после переключения стека весьма возможно, что он будет указывать куда-то внутрь стража и существует (хотя и весьма незначительная) вероятность, что страж будет испорчен до того, как успеет вернуть управление (например, таймерным прерыванием). Для решения этой проблемы можно использовать следующий факт: любая команда пересылки в регистры SS и SP запрещает прерывания на время выполнения следующей команды. Таким образом, возврат управления из стража должен завершаться следующими командами:

```
MOV SS,...  
MOV SP,...  
JMP ...
```

5. Как лечить защищенную программу, если она заражена. Страж не защищает программы от заражения. Он лишь максимально своевременно его диагностирует и сообщает о нем.

А как вылечить программу от вируса? Для этого необходима определенная информация о здоровом состоянии программы. Эта информация включает в себя длину незараженного файла: в случае COM файла - первые несколько байтов этого файла (в моем страже - 512 байт), а в случае EXE файла - первые несколько байтов его заголовка (в моем страже - 28 байт). Маловероятно, чтобы вирус испортил что-нибудь еще, однако не пренебрегайте и такой неприятной возможностью (в этом случае файл вылечить нельзя). Чтобы выявить неизлечимые файлы, необходимо внутри стража иметь какую-нибудь контрольную сумму тех частей файла, копии которых не хранятся внутри него. А что если у зараженного файла эта контрольная сумма изменилась? Тогда весьма вероятно, что вирус испортил какие-то его незащищенные части и вылечить файл нельзя. Хотя более вероятно, что его настраивали, инсталлировали и т.п., что и внесло в него эти изменения. Даже если контрольная сумма изменилась, все равно имеет смысл попробовать восстановить файл, а затем его запустить (если начал работать, значит, вылечился!).

6. Чего все это стоит? Лично я написал стража за одну неделю не слишком напряженной работы. Объем реализации - 1650 строк на Си и 125 строк на ассемблере.

Соображения о загрузочных вирусах

Загрузочный вирус характерен тем, что изменяет (портит) загрузочный сектор диска. Следовательно, необходимо достаточно часто сравнивать загрузочный сектор с эталоном и в случае несовпадения заменять испорченный загрузочный сектор эталонным. Столь прямолинейно применить общую идеологию, к сожалению, невозможно. Трудности, с которыми придется столкнуться, различны для жестких (винчестерских) и гибких дисков. Рассмотрим их отдельно.

Защита гибких дисков от загрузочных вирусов

Хранить на каждой дискете эталон ее загрузочного сектора довольно затруднительно - дискеты непрерывно кочуют с места на место. Это означает, что к пользователю регулярно приходят дискеты извне. Естественно, что на них никто не поместил эталон загрузочного сектора, с них регулярно стирают все содержимое (например, путем форматирования) и т.п. Короче говоря, требовать наличия эталонной копии загрузочного сектора на каждой дискете - это было бы слишком жестоко по отношению к пользователю (ведь антивирусные программы ни в коем случае не должны стеснять его свободу).

Кроме того, не совсем понятно, в какой момент естественно сверять загрузочный сектор с эталоном. Если при каждом обращении к диску, то это слишком часто, а если только по прямому запросу пользователя, то может быть и слишком редко (ведь пока гром не грянет, мужик не перекрестится).

Я предлагаю при решении проблемы исходить из таких соображений:

1. Поскольку полностью контролировать зараженность каждой дискеты невозможно, то и пытаться это делать ни к чему. Вместо этого необходимо уделить внимание диагностике заражения жесткого диска на ранней стадии и эффективным методам лечения как жесткого диска, так и дискет.

2. Необходима эффективная профилактическая процедура для тех "мужиков, которые привыкли креститься регулярно". Такую процедуру я реализовал в виде несложной программы, которая заменяет загрузочный сектор любой дискеты на заведомо незагрязненный. Эта же программа возвращает нормальные кластеры (помеченные как дефектные) обратно в число нормальных и протирает нулями все пустые ячейки корневой директории. В результате все бывшие на дискете файлы оста-

ются нетронутыми, а вирус (если он был) удаляется. Подобной чистке я подвергаю всякую дискету, которая побывала в подозрительной ПЭВМ или которую я собираюсь кому-то отдать. Недостаток этой процедуры в том, что загружаемую дискету (например, дискету с ОС или самозагружаемой игрушкой) она делает незагружаемой. Но в игрушки я не играю, а в остальных случаях потребность в загружаемой дискете возникает довольно редко. Конечно, эффективной профилактической процедурой может служить форматирование дискеты, но, с моей точки зрения, оно осуществляется слишком медленно и к тому же не сохраняет записанных на дискету данных.

Защита жестких дисков от загрузочных вирусов

С жестким диском, казалось бы, все ясно. Хранить эталонную копию загрузочного сектора в специальном файле? Такое вполне можно себе позволить. Сверять загрузочный сектор с эталоном уместно при начальной загрузке компьютера. Это можно сделать совершенно незаметно для пользователя - достаточно поместить соответствующую программу в AUTOEXEC.BAT. Однако здесь нас подстерегает серьезная трудность. Чтобы разобраться в ее природе, нам придется ознакомиться с некоторыми сведениями о способе доступа к диску.

Но прежде упомяну такой малоизвестный факт: на жестком диске загрузочных секторов - два.

Один из них - MBR (Master Boot Record) - находится в самом начале диска (первый сектор нулевой дорожки нулевого цилиндра). Его нельзя посмотреть с помощью рctools, ну и т.п. Прочитать и записать его все же можно с помощью INT 13h, но об этом ниже.

Другой загрузочный сектор (вторичный) находится в самом начале активного раздела жесткого диска. Раздел - та часть диска, которой присвоена какая-либо буква (C:, D: и т.п.). После загрузки DOS (на момент начала выполнения файла AUTOEXEC.BAT) обычно активен раздел C:. Именно вторичный загрузочный сектор можно посмотреть утилитами ну, рctools и т.п. Писать и читать вторичный загрузочный сектор также можно с помощью INT 13h, но проще это делать с помощью INT 25h/26h.

Хоть бы один из этих загрузочных секторов был неуязвим для вирусов. Так нет же! Оба участвуют в начальной загрузке, оба могут быть заражены вирусом (вирус "Marijana" заражает MBR, а "Итальянский попрыгунчик" - вторичный загрузочный

сектор). Таким образом, необходимо иметь эталонные копии обоих этих секторов.

Как проходит чтение жесткого диска

Все программы, работающие с диском, делают это, обращаясь к специальной программе (INT 13h) с различными параметрами. Название INT 13h означает, что адрес этой программы лежит в специальной ячейке памяти: вектор прерывания номер 13h, или, сокращенно, вектор INT 13h (буква h в конце числа означает, что оно шестнадцатеричное; в десятичной системе счисления 13h равно 19). В момент начальной загрузки в векторе INT 13h лежит адрес программы, находящейся в ПЗУ компьютера. Эта программа - часть так называемого BIOS - базовой системы ввода-вывода. Затем, по мере загрузки DOS, некоторые ее компоненты (а именно сама DOS и некоторые загружаемые драйверы, например DMDVR и др.) перехватывают INT 13h:

- 1) содержимое вектора INT 13h считывает и запоминает перехватывающая компонента;

- 2) в вектор INT 13h записывается адрес некоторой программы, находящейся внутри перехватывающей компоненты.

Теперь при всяком обращении к INT 13h управление передается той компоненте DOS, которая последней перехватила INT 13h. Эта компонента проводит некоторые специфичные для нее действия, в процессе обращаясь к той программе, адрес которой запомнен внутри компоненты как адрес "настоящего" INT 13h. На самом деле это может быть не настоящий INT 13h, а другая, перехватившая его компонента, и т.д. по цепочке. В конце концов управление получит программа, находящаяся в ПЗУ, адрес которой содержался в векторе INT 13h в момент начальной загрузки. Не исключено, что этой последней программе будут переданы не совсем те параметры, которые передавались при вызове INT 13h.

Допустим, что мы хотим, например, считать MBR и с этой целью вызываем INT 13h, передавая ей соответствующие параметры (т.е. читать надо первый сектор нулевой дорожки нулевого цилиндра). На самом деле у нас нет никакой гарантии, что считан будет именно MBR, ведь программа, перехватившая INT 13h, вполне могла подменить параметры и считать совсем другой сектор. Конечно, при нормальной работе ПЭВМ ничего подобного не происходит, так как все программы, перехватившие INT 13h, ведут себя "честно". Но когда

компьютер заражен загрузочным вирусом, подобный “обман” вполне возможен.

Зачем загрузочный вирус перехватывает INT 13h...

Итак, предположим, что компьютер заражен загрузочным вирусом. В момент заражения вирус скопировал загрузочный сектор жесткого диска (какой-то из двух) в “безопасное место” и записал на его место зараженный загрузочный сектор. Теперь в момент начальной загрузки с жесткого диска вместо обычного загрузочного получит управление зараженный сектор. Этот зараженный загрузит вирус в оперативную память. Загруженный вирус немедленно перехватывает INT 13h. (Хотя ни откуда логически не следует, что он обязан это сделать, но все известные мне загрузочные вирусы так себя ведут и у них есть на то важные причины. Одной из причин мы коснемся ниже.) После этого вирус передает управление обычному загрузочному сектору (ведь у него в “безопасном месте” есть его копия). Далее загрузка продолжится, как обычно.

Что теперь будет, если некоторая программа хочет считать загрузочный сектор и с этой целью вызывает INT 13h, передав ему соответствующие параметры? Ради самомаскировки вирус, перехвативший INT 13h, подменит передаваемые параметры и вызовет чтение копии нормального загрузочного сектора из “безопасного места”. То же произойдет при попытке записи загрузочного сектора, т.е., во-первых, мы не узнаем, что загрузочный сектор заражен, а во-вторых, узнав (например, увидев шарик, бегающий по экрану), ничего не сможем сделать. Точнее сказать, сможем, если загрузимся с защищенной незараженной системной дискеты. У Вас ведь, конечно, есть такая дискета? Вот почему трудно обнаружить загрузочный вирус.

... и как с этим бороться

Единственный способ прочесть именно загрузочный сектор, а не что-либо другое - это непосредственно обратиться к той программе в ПЗУ, адрес которой записан в векторе INT 13h в момент начала загрузки. Но для этого надо знать ее адрес, а узнать его не так-то просто.

Действительно, чтобы его узнать, необходимо запустить какую-то программу. Программу можно запустить только по-

сле того, как загружена DOS. Но к этому моменту INT 13h уже перехвачен (может быть неоднократно) и вектор INT 13h содержит адрес какой-то программы в какой-то компоненте DOS или где-нибудь еще (например, INT 13h мог перехватить ANTI4US). Простого легального способа узнать адрес этой программы из ПЗУ нет. Среди не очень простых и не очень легальных наиболее привлекателен способ, применяемый в отладчиках, а также в некоторых вирусах и антивирусах. Это трассировка.

В чем суть трассировки? Некоторая программа (трассирующая) выполняет другую программу (трассируемую) команда за командой, совершая после выполнения каждой команды трассируемой программы некоторые действия (например, проверку некоторых условий, запоминание последовательности выполняемых команд и т.п.). В таких случаях говорят, что первая программа (трассирующая) трассирует вторую программу (трассируемую).

Итак, если мы будем трассировать INT 13h, то в конце концов придем к программе, записанной в ПЗУ, и узнаем ее адрес. Назовем эту программу первичной. Более того, если трассирующая программа достаточно интеллектуальна, то можно узнать даже адрес той ячейки, в которой лежит адрес программы, записанной в ПЗУ. Назовем эту ячейку корневой.

Теперь, когда мы знаем адрес программы в ПЗУ и адрес корневой ячейки, их надо запомнить и впоследствии проверять зараженность загрузочным вирусом так:

1) проверить, лежит ли в корневой ячейке адрес первичной программы. Если нет, то, скорее всего, компьютер заражен загрузочным вирусом. В этом случае надо положить в корневую ячейку адрес первичной программы (см. замечание в п.3 ниже);

2) поскольку теперь обращение к INT 13h не приводит к обращению к вирусу, то можно спокойно читать загрузочные секторы (MBR с помощью INT 13h, а вторичный с помощью INT 25h) и сравнивать их с эталонами;

3) если есть расхождение с эталоном, то можно, совершенно не опасаясь вируса, записать эталонный загрузочный сектор на место зараженного.

З а м е ч а н и е. Прежде, чем делать это, необходимо спросить разрешение у пользователя. Действительно, ведь расхождение с эталоном могло произойти и потому, что пользователь изменил распределение разделов или перешел на новую ОС или просто перенес эту программу на другой компьютер, забыв создать новый эталон загрузочных секторов. Это замечание относится и к п.1 (ведь адрес корневой ячейки может из-

мениться при смене ОС, а адрес первичной программы - при смене ПЭВМ). Вопрос пользователю надо сформулировать внятно, посоветовав ему ничего не предпринимать, если он в чем-то не уверен. Действительно, никакой срочности ведь нет - удалим вирус не сегодня, так завтра, посоветовавшись с местными авторитетами. Вообще, любые антивирусные действия следует начинать без спешки и паники, на свежую голову.

Заметим, что после проверки и лечения таким образом нам даже не потребуется загружать ПЭВМ с незараженной системной дискеты (но иметь ее все равно необходимо). После того, как загрузочные секторы восстановлены, надо перезагрузить ПЭВМ, и вирус будет удален из памяти. Удалив вирус с жесткого диска, необходимо удалить его и со всех дискет, подозреваемых в том, что они побывали в зараженном компьютере. Это можно сделать, например, с помощью форматирования или моей программой, о которой говорилось при обсуждении защиты дискет от загрузочных вирусов.

Кое-что о трассировке

Самое сложное в предлагаемом проекте - трассировка. Сам я в целях борьбы с загрузочными вирусами неоднократно трассировал INT 13h с помощью отладчика AFD, легко находя и адрес первичной программы, и адрес корневой ячейки. Однако автоматическая трассировка - процедура более тонкая, включающая в себя элементы "неестественного интеллекта". Никаких разумных письменных источников по этому вопросу я не нашел, поэтому считаю необходимым рассмотреть некоторые технические моменты.

В слове состояния процессора (PS) есть бит (8-й), называемый Т-битом. Когда Т-бит установлен в 1, после выполнения почти каждой машинной команды (исключения см. ниже) возникает прерывание INT 1. Этот факт - основа всех трассирующих программ. Установить Т-бит в 1 проще всего с помощью машинной команды IRET. Отметим, что прерывание INT 1 не возникает после выполнения машинной команды INT. Кроме того, флаг может быть сброшен машинными командами IRET и POPF.

Известен вирус (правда, не загрузочный), который отлавливает трассировку себя и препятствует ей. Во избежание подобного на время трассировки рекомендуется запретить прерывания от таймера (или вообще все прерывания), а чтобы это не мешало работе INT 13h, - трассировать какую-нибудь простейшую функцию (подходящи для этого функции

получения параметров дисководов и кода последней ошибки). Кроме того, не советую во время обработки прерывания INT 1 вызывать программу, адрес которой раньше лежал в векторе INT 1 (т.е. в отличие от INT 13h или .п. обработчиков прерывания INT 1 не рекомендуется связывать в цепочки).

Как при трассировке выяснить первичный и корневой адреса? С большой долей уверенности можно считать первичным адрес (CS:IP) в тот момент, когда CS впервые начал указывать на ПЗУ. Команда, передавшая управление на ПЗУ, будет, скорее всего, косвенной командой CALL, JMP либо командой INT. Во всех этих случаях адрес корневой ячейки содержится в коде команды. Определенную неприятность может представлять случай, когда в момент трассировки компьютер уже заражен вирусом. Но этот случай легко "отловить", проверяя, не указывает ли адрес (CS:IP) на область, уже недоступную для DOS, но еще не являющуюся ПЗУ (именно так "живут" вирусы). Если указывает, - компьютер уже заражен. Его надо лечить другими средствами.

К моменту публикации статьи антивирус BG, защищающий от всех загрузочных вирусов, работает. Он излечивает компьютер всегда, если в момент установки антивируса компьютер не был инфицирован, и почти всегда, если был.

Телефон 939-17-86. Москва

УДК 681.3.06

В.Ю.Баевский

КОНФЛИКТ BETA И LANSMART. КАК ИХ ПРИМИРИТЬ, НЕ БЕСПОКОЯ А. ЧИЖОВА?

1. BETA и LANsmart

Кто не знает систему Beta маэстро А.Чижова для русификации IBM PC-совместимых компьютеров? Обилие драйверов, поддерживающих все мыслимые и немыслимые клавиатуры, мониторы и принтеры, сделали Beta одним из самых распространенных в нашей стране системных программных средств для IBM PC, наряду с MS-DOS и Norton Commander. Мне мо-

гут возразить, что есть и другие русификаторы. Согласен, бродят по компьютерам EGAGA, KBRM, RK и другие, по мнению их авторов, более совершенные кириллические драйверы. Но это все новации, экспериментальные модели, так сказать. А вот Beta - это, простите, классика, как старина NC или PE2. А кто же отринет классику? Так что давайте сойдемся на том, что Beta популярна и останется таковой в обозримом будущем.

Кто не знает сетевую ОС LANsmart фирм D-Link/Datex inc.? Думаю, что таких окажется чуть-чуть побольше. Ведь этим средством пользуются, в основном, владельцы сетевых карт фирмы D-Link, а также владельцы SMC-совместимых карт ARCnet, предпочитающие дорогому и тяжеловесному сетевому сервису NetWare дешевизну и разумную достаточность LANsmart. Для тех пользователей IBM PC, кто не знаком с LANsmart, приведу о нем некоторые сведения. Может быть, они захотят его приобрести.

LANsmart - это многопользовательская сетевая ОС. Она в ряде приложений успешно конкурирует со своим именитым соплеменником (NetWare фирмы Novell), но отличается от последнего дешевизной и компактностью. Стоимость LANsmart 2.XX - менее 400 долл., а дистрибутив занимает всего одну дискету формата 360 Кбайт (сравните с NetWare 286 Advanced 2.15, стоимость которого больше 3000 долл., а дистрибутив занимает почти 40 дискет). Опыт работы автора в СП "Ассоциация АМИКСТ" убеждает в том, что довольно часто LANsmart полностью устраивает пользователя во всех аспектах.

Что же может LANsmart? Не так уж мало! Обслуживать локальную сеть из нескольких десятков компьютеров. Предоставлять ресурсы серверов (жесткие диски и принтеры) любым компьютерам, входящим в сеть (как рабочих станций, так и серверов). Обеспечивать удобный доступ к этим ресурсам по именам. Она предусматривает парольную защиту, дает возможность достаточно эффективно работать на сервере как на рабочей станции (это неудобно либо невозможно при работе с NetWare). Позволяет динамически менять сервер на рабочей станции и наоборот (что невозможно в NetWare). Поддерживает все сетевые карты фирмы D-Link (карты Ethernet - DE-001, DE-100, DE-200 и ARCnet - DA-100, DA-200), а также, как уже сказано, все SMC-совместимые карты ARCnet. И за все это LANsmart берет всего 60 Кбайт оперативной памяти на рабочей станции, а на сервере - почти 140 Кбайт.

"Что же общего между Beta и LANsmart, какая между ними связь?" - недоуменно спросит читатель. Ровным счетом ни-

чего общего у них нет. Они не совместимы* и не могут, к сожалению, работать вместе на IBM PC. По-моему, требуется вмешательство программиста, но об этом ниже.

2. Суть проблемы

Да, уважаемый читатель, Beta и LANsmart (в отдельности очень неплохие программные продукты), будучи запущены на одном компьютере, способны лишь вызвать жесткий цикл, от которого спасает только кнопка RESET. Причины этой неполадки - несовершенство дисциплины обновления и использования вектора 2FH. Этот вопрос, как кажется автору, представляет самостоятельный интерес. Имеет смысл остановиться на нем более подробно.

Так уж сложилось традиционно в IBM PC, что замена содержимого векторов стала обыденным приемом, используемым системными и прикладными программными средствами**. Причем изменить содержимое даже жизненно важных векторов долгое время можно было без какой-либо дисциплины. С версией 3.20 MS-DOS введено понятие так называемого мультиплексируемого вектора (Mux vector) [1,2]. Для него характерна некоторая дисциплина использования, дающая, по мнению ее авторов, возможность устанавливать вектор многими процессами***. Таким вектором стал 2FH (0000:00BCH), ранее применяемый только для печати в фоновом режиме. В чем состоит введенная дисциплина? Принципы ее таковы. Каждый процесс может установить свою собственную резидентную часть (прогамму), характеризующую числовым идентификатором от 0 до 255. Адрес резидентной части заносится в вектор 2FH, а оригинальное содержимое вектора сохраняется. Установленные друг за другом резидентные части образуют, таким образом, цепочку. При этом резидентная часть, установленная последней, при прерывании 2FH получит управление первой. Если запрос адресован не ей, то управление передается резидентной части другого процесса, установленной ранее, и т.д. (рис.1). Обращаясь к своей резидентной части через вектор 2FH, процесс обязан занести в регистр AH свой идентификационный номер, который и будет опознан требуемой резидент-

* Автор может сказать это с уверенностью только о широко распространенных версиях Beta.

** Речь идет, разумеется, о работе под управление MS-DOS в незащищенном режиме.

*** Терминология MS-DOS.

ной частью. Каждая резидентная часть может выполнять некоторый набор функций, любая из которых характеризуется целым числом от 0 до 255. При обращении к своей резидентной части процесс должен занести код функции в регистр AL. Функция 0 резервируется для определения состояния резидентной части. В ответ на запрос функции 0 возможны два ответа в регистре AL: FFH - резидентная часть установлена, 0 - резидентная часть не установлена. Если резидентная часть уже установлена, процесс может обращаться к ней за выполнением тех или иных функций, если нет, то он должен ее установить.

Нетрудно заметить (на это справедливо указано в [1,2]): при такой дисциплине работы с вектором 2FH теоретически всегда есть опасность, что несколько процессов изберут (разумеется, по воле их авторов) одинаковые идентификационные номера, а это приведет к путанице и авариям.

Вектор 2FH

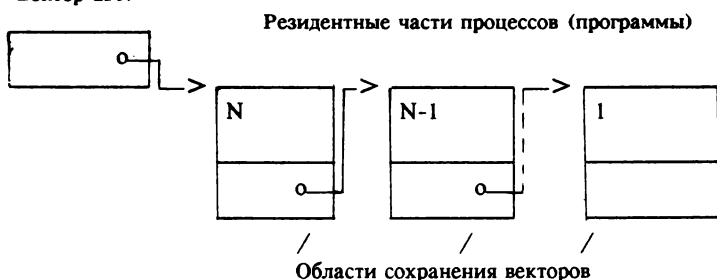


Рис.1 Цепочка из N резидентных частей

Исследования с помощью разработанного для этой цели трассировщика обращений к вектору 2FH показали - эта теоретическая возможность замечательно осуществляется практически в случае Beta и LANsmart. Дело в том, что оба средства используют идентификационный номер EDH. Вот почему, если сначала установлена Beta, LANsmart ошибочно решает, будто его резидентная часть уже установлена, и обращается к ней через вектор 2FH (это ведет к жесткому циклу). А если сначала запущен LANsmart и затем вызывается Beta? Также возникает конфликт, так как Beta не проверяет установленность, а сразу модифицирует вектор (опять-таки цикл при работе LANsmart!)

Наиболее радикальным было бы переделать Beta или LANsmart. Но для этого пришлось бы побеспокоить А.Чижова - творца Beta, либо фирму D-Link/Datex inc. Можно было бы,

правда, модифицировать непосредственно объектный код одного из этих средств. Это, однако, и трудоемко, и ненадежно, и некорректно по отношению к авторам. К счастью, нашелся простой способ преодоления этой проблемы, свободный от перечисленных недостатков.

3. Решение проблемы - программа LANBeta

Вектор 2FH используется Beta только на этапе загрузки драйверов. Это позволило создать простую резидентную программу-посредник, которой дано не слишком оригинальное название LANBeta. В чем идея примирения Beta и LANsmart? Изолировать LANsmart от Beta перехватом запроса с кодом идентификации EDH и кодом функции 0. Это достигается установкой LANBeta сразу же после установки Beta. В ответ на соответствующий запрос LANBeta возвращает в регистре AL код, отличный от FFH. Это побуждает LANsmart установить свою резидентную часть, после чего он будет получать управление до LANBeta по любым запросам с кодом идентификации EDH, а LANBeta вообще перестанет активно функционировать. Однако свою роль примирителя она к этому моменту уже сыграет.

Полный текст LANBeta приведен на рис.2. LANBeta ориентирована на любой IBM PC-совместимый компьютер, оснащенный EGA. Разумеется, ее легко сделать не зависимой от монитора.

4. Как успешно применять LANBeta?

Для этого необходимо гарантировать ее запуск сразу же после системы Beta. С этой целью можно было бы включить в файл AUTOEXEC.BAT фрагмент, например, вида:

```
...
beta
betaload  rn~ibmss an~qwe.kbl
betaload  rn~epsfx.pr an~l208.prf
betaload  rn~ega.sc an~0814.scf rn~bord.sce
lanbeta
...
```

Однако этого недостаточно. Дело в том, что если пользователь, работая с LANsmart, пожелает сохранить конфигурацию рабочей станции или сервера, то произойдет следующее. Файл AUTOEXEC.BAT будет переименован в

AUTOUSER.BAT, затем будет создан новый AUTOEXEC.BAT, куда войдут команды запуска LANsmart и установки требуемой конфигурации с вызовом AUTOUSER.BAT в конце файла. В результате требуемая последовательность запуска Beta, LANBeta и LANsmart будет нарушена. Чтобы этого не произошло, сохранив конфигурацию, перед следующей перезагрузкой DOS следует из файла AUTOUSER.BAT перенести строки, относящиеся к Beta и LANBeta в начало нового AUTOEXEC.BAT. Перед последующими сохранениями конфигурации необходимо вернуть эти строки из AUTOEXEC.BAT в AUTOUSER.BAT (в противном случае LANsmart выдаст сообщение об ошибке). Разумеется, необходимость таких “ручных” операций неприятна, но, к счастью, как показывает практика, выполнять их приходится редко. Есть и другие способы обеспечить необходимую последовательность вызовов, но для этого требуется написать дополнительные программы.

CSEG	ASSUME SEGMENT ORG	CS:CSEG BYTE 100H	
LANBeta	PROC	FAR	
HAND:	JMP CMP JNE CMP JE	INSTALL AH,0EDH GOPREV AL,0 NOTINS	: Идти на установку : Конфликт идентификаций? : Нет : Текст на установленность? : Да
GOPREV:	JMP	DWORD PTR CS:OLDVECT	: Отдать управление предыдущему : обработчику
NOTINS:	MOV IRET	AL, 01H	: Сообщить, что резидентная часть не : установлена : Отдать управление запросившему : процессу
OLDVECT	DW	2 DUP (?)	:
ENDRES	LABEL	BYTE	: Конец резидентного кода
INSTALL:	MOV MOV INT MOV LEA MOV MOV INT MOV INC INT	AH, 3 BH, 0 10H AX, 1300H BP, MSG BX, 000FH CX, MSGEND-MSG 10H AN, 2 DH 10H	: Прочсть текущую позицию курсора : Номер видеостраницы : : DH,DL содержат строку и колонку : EGA сервис (вывести строку, не : обновляя курсора) : Адрес сообщения : Видеостраница 0, ярко-белая на черном : фоне : Длина сообщения : : Продвинуть строку : :

```

MOV     AX, 352FH           ; Получить и сохранить содержимое
INT     21H                 ; вектора 2FH

MOV     CS:OLDVECT, BX      ;
MOV     CS:OLDVECT + 2, ES;

MOV     AX, 252FH           ; Установить собственный вектор
PUSH    CS                  ;
POP     DS                  ;
MOV     DX, OFFSET HAND ;
INT     21H                 ;

MOV     DX, OFFSET ENDRES;  Длина резидентного кода
INT     27H                 ; Завершиться и остаться резидентным

```

```

MSG DB DIMAsoft (R) LANBeta - LANsmart & Beta reconciler (C) 1990', 7
MSGEND EQU $

```

```

LANBeta   ENDP

CSEG      ENDS
          END      LANBeta

```

Рис.2. Полный текст программы-примирителя LANBeta

ЛИТЕРАТУРА

1. *TECH Help Text, Version 3.2a, 1985.*
2. *TECH Help Text, Version 3.3a, 1986.*

УДК 681.322.1

А. А. Некрасов

ПОДГОТОВКА ВИНЧЕСТЕРСКОГО ДИСКА К РАБОТЕ

Этот материал был прочитан автором в виде лекции для специалистов, проводящих сервисное обслуживание ПЭВМ семейства IBM PC

Подготовка винчестерского диска (Hard Disk) к работе заключается в разбиении его на необходимое число индивидуальных логических областей (Partitions), форматировании и

постановке на него ОС (или нескольких ОС) и прикладного ПО пользователя.

Практика эксплуатации ПЭВМ показывает, что пользователю рано или поздно приходится разбивать винчестерский диск на Partitions и форматировать его, например при установке нового или дополнительного накопителя.

Такие операции в MS-DOS выполняются специальными утилитами. Наиболее распространена в стране версия MS-DOS 3.30, поэтому в статье рассматривается утилита этой версии ОС. Как упоминалось, Hard Disk можно разбить на логические части (Partitions). Partitions образуют на винчестерском диске индивидуальные области, которые могут содержать ОС или быть совместимыми с нею.

Утилита FDISK используется для:
создания первичной (Primary) или расширенной (Extended) MS-DOS Partitions;
изменения активной (Active) Partitions;
уничтожения MS-DOS Partitions;
вывода информации о Partitions на дисплей;
просмотра или модификации конфигурации второго винчестерского диска.

Внимание! Изменить конфигурацию диска - значит потерять существующие на нем файлы. Поэтому перед началом работы с FDISK скопируйте все файлы с винчестерского диска на промежуточные носители.

Подготовительные операции

Многие ПЭВМ поступают от изготовителя с винчестерскими дисками, подготовленными для работы под MS-DOS. В этом случае отпадает необходимость в использовании FDISK. Чтобы узнать, так ли это, попытайтесь запустить MS-DOS с винчестерского диска.

Если винчестерский диск разбит на Partitions и отформатирован, а системные файлы перенесены на него, то MS-DOS загрузится.

Если MS-DOS не загружается, а ваша ПЭВМ совместима с IBM PC/AT, проверьте, какой тип накопителя указан в (C)MOS-памяти. Для этого необходимо запустить программу Setup, определяющую конфигурацию ПЭВМ. В старых моделях ПЭВМ программа Setup находится на дискете с MS-DOS, а в новых моделях - в ПЗУ (вызывается одновременным нажатием клавиш Alt-Ctrl-Esc или Del после включения машины

перед загрузкой ОС). Если тип накопителя не указан или указан неправильно, запишите в (C)MOS-память тип винчестерского диска, используемого вами. Например, для 286 Modular BIOS версии 3.03HL (Award Software Inc.) винчестерский диск Seagate ST-251 (40 Мбайт, 820 цилиндров, 6 головок, 17 секторов) имеет тип 40. Интересно, что для различных версий Setup типы одного и того же винчестерского диска могут не совпадать. Затем проверьте конфигурацию диска с помощью утилиты FDISK (раздел "Display Partitions Information").

Если MS-DOS Partitions существуют, то диск был сконфигурирован, если нет, то необходимо их создать (см. ниже).

Как запустить FDISK?

Утилита FDISK использует меню при проведении каждой процедуры и поэтому очень проста в работе. Для запуска FDISK необходимо: поместить дискету с MS-DOS в дисковод A; перезагрузить компьютер; набрать на клавиатуре FDISK и нажать на клавишу "Enter". На экране дисплея появится основное меню:

Fixed Disk Setup Program Version 3.30
(C) Copyright Microsoft Corp. 1987

FDISK Options

Current Fixed Disk Drive: 1

Choose one of the following:

1. Create DOS Partition
2. Change Active Partition
3. Delete DOS Partition
4. Display Partition Data
5. Select Next Fixed Disk Drive

Enter choice: [1]

Press ESC to return to DOS

Это меню может содержать пять функций. Если у ПЭВМ только один Hard Disk, то пятая функция на экране не отобразится.

Ниже будут описаны все функции основного меню, показаны их опции и информация, выдаваемая на дисплей. Для возврата в MS-DOS из основного меню нажмите на клавишу ESC. Клавиша ESC используется также для возврата в основное меню из любых опций FDISK.

В строке “Выбор” стоит по умолчанию опция 1. Для выбора этой опции необходимо нажать на клавишу “Enter”. Для выбора другой опции наберите на клавиатуре ее номер, а затем нажмите “Enter”.

Как получить информацию о Partition?

При вызове опции 4. Display Partition Information основного меню FDISK высвечивает сообщение с информацией о каждой Partition на винчестерском диске:

Display Partition Information

Current Fixed Disk Drive:1

Partition	Status	Type	Start	End	Size
C:1	A	PRI DOS	0	409	410
2		EXT DOS	410	818	409

Total disk space is 819 cylinders.

The Extended DOS Partition contains logical DOS driver. Do you want to display logical drive information? [Y]

Press ESC to return to FDISK Options

Это информационное окно показывает номер каждой Partition, статус, тип, начальный и конечный цилиндры и размер (в цилиндрах). Если у вас есть Extended DOS Partition, то FDISK спросит: Хотите ли вы увидеть данные о логических дисках? Наберите “Y” и нажмите “Enter”. На экране будет:

Display Logical DOS Drive Information

DRV	Start	End	Size
D:	410	613	204
E:	614	818	205

Press ESC to return to FDISK Options

Как создать DOS Partitions?

При выборе первой опции основного меню на дисплей выводится информация:

Create DOS Partition

Current Fixed Disk Drive: 1

- 1. Create Primary DOS Partition**
- 2. Create Extended DOS Partition**

Enter choice: [1]

Press ESC to return to FDISK Options

Рассмотрим, как создается Primary DOS Partition.

Она должна быть создана раньше, чем Extended DOS Partition. Для этого нажмите "Enter" при выборе опции 1. На дисплее появится сообщение:

Create Primary Dos Partition

Current Fixed Disk Drive: 1

**Do you wish to use the maximum size
for DOS Partition and make the DOS
Partition active (Y/N) ...? [Y]**

Press ESC to return to FDISK Options

Для использования максимального объема диска под Primary DOS Partitions нажмите "Enter" при выборе [Y]. Окончив работу, FDISK выведет на дисплей сообщение:

System will now restart

**Insert DOS diskette in drive A:
Press any key when ready...**

Пользователю необходимо вставить дискету с MS-DOS в дисковод А и нажать на любую клавишу для загрузки MS-DOS.

Теперь, когда создана MS-DOS Partition, отформатируем винчестерский диск. Чтобы MS-DOS загружалась с него, надо использовать ключ /s в программе **format**, а именно:

```
format c:/s
```

Чтобы создать Primary DOS Partition размером меньше максимально возможного, необходимо нажать клавишу [N] при ответе на вопрос предыдущего меню. FDISK выдаст на дисплей следующее меню. В нем укажите Primary DOS Partition:

Create Primary DOS Partition

Current Fixed Disk Drive: 1

Total disk space is 819 cylinders

**Maximum space available for partition
is 642 cylinders**

. Enter partition size....: [642]

No partitions defined

Press ESC to return to FDISK Options

Введите с клавиатуры желаемый размер Partition (в цилиндрах) и после этого нажмите "Enter". FDISK создаст Primary DOS Partition и сообщит:

Create Primary DOS Partition

Current Fixed Disk Drive: 1

Partition	Status	Type	Start	End	Size
C: 1		PRI DOS	0	409	410

Primary DOS Partition created

Press ESC to return to FDISK Options

Созданная Partition не активная. Оставшийся объем диска можно использовать для Extended DOS Partition.

Создание Extended DOS Partition

Если объем диска превышает 32 Мбайт (максимально возможный размер Primary DOS Partition) или если пользователю нужно более одного логического диска на одном накопителе, то с помощью FDISK можно создавать Extended DOS Partition. Для этого надо выбрать опцию Create Extended DOS Partition, набрав на клавиатуре "2", затем нажать "Enter". В ответ FDISK сообщит:

Create Extended DOS Partition

Current Fixed Disk Drive: 1

Partitions	Status	Type	Start	End	Size
C:1	A	PRI DOS	0	409	410

Total disk space is 819 cylinders.
Maximum space available for partition
is 409 cylinders

Enter partition size....: [409]

Press ESC to return to FDISK Options

Это меню показывает общее количество цилиндров, доступных для Extended DOS Partition. Ее размеры ограничивает максимально доступный объем винчестерского диска. Если вам необходимо создать Extended DOS Partition размером, меньшим максимально допустимого, наберите на клавиатуре размер Partition (в цилиндрах) и нажмите "Enter". На экране появится сообщение:

Create Extended DOS Partition

Current Fixed Disk Drive: 1

Partitions	Status	Type	Start	End	Size
C: 1	A	PRI DOS	0	409	410
2		EXT DOS	410	509	100

Extended DOS Partition created

Press ESC to return to FDISK Options

Создание логических дисков в Extended DOS Partition

Внутри Extended DOS Partition нужно выделить один или несколько логических дисков. FDISK автоматически высвечивает на дисплее меню для их создания:

Create Logical DOS Drive(S)

No logical drives defined

Total partition size is 409 cylinders.
Maximum space available for logical
drive is 409 cylinders.

Enter logical drive size....: [409]

Press ESC to return to FDISK Options

Всю Extended Partition можно организовать как один, два (или более) логических диска. Единственное ограничение - размер логического диска не должен превышать 32 Мбайт.

После того, как вся Partition разбита на логические диски, FDISK сообщит:

Create Logical DOS Drive(S)

Drv	Start	End	Size
D:	410	613	204
E:	614	818	205

All available space in the Extended DOS
Partition is assigned to logical drives.

Logical DOS drive created, drive letters
changed or added

Press Esc to return to FDISK Options

В результате весь доступный объем в Extended DOS Partition разбит на логические диски.

Нажмите на ESC для возврата в основное меню FDISK. После этого можно перезагрузить ПЭВМ или выбрать другую опцию.

Как изменить статус Partition?

При выборе в основном меню FDISK опции 2. **Change Active Partition** на экране появляется сообщение о статусе каждой Partition на винчестерском диске. Активная Partition обозначается буквой A. Если на диске, кроме того, есть Partition с другой ОС, то меню позволяет сделать ее активной. Только одна Partition может быть активной в текущий момент.

Если на диске будут и Partitions XENIX, и DOS, изменить активную Partition можно так:

Change Active Partition

Current Fixed Disk Drive: 1

Partition	Status	Type	Start	End	Size
c: 1		non DOS	0	409	410
2	A	PRI DOS	410	818	409

Total disk space is 819 cylinders.

Enter the number of the partition you
want to make active...: [1]

Press ESC to return to FDISK Options

Наберите номер Partition, которую надо сделать активной, и нажмите "Enter". Активной станет Partition с выбранным номером.

Если у вас на винчестерском диске 1 созданы Primary DOS Partition и Extended DOS Partition, то нельзя сделать активной Extended DOS Partition. В этом случае FDISK сообщит:

**Partition selected (2) is not bootable,
active partition not changed**

Как уничтожить DOS Partition?

Опция 3. **Delete DOS Partition** основного меню FDISK выдает пользователю сообщение с вопросом, какую Partition необходимо удалить - Primary или Extended:

Delete DOS Partition

Current Fixed Disk Drive: 1

Choose one of following:

- 1. Delete Primary DOS Partition**
- 2. Delete Extended DOS Partition**

Enter choice: []

Press ESC to return to FDISK Options

Если Extended DOS Partition содержит несколько логических дисков, в приведенном меню добавится еще один пункт:

- 3. Delete Logical DOS Drive(S) in
the Extended DOS Partition**

Набрав на клавиатуре номер выбираемой опции, нажмите "Enter": FDISK удалит Partition, и все данные, которые там есть, будут уничтожены. Однажды удалив Partition, невозможно восстановить данные, которые были там записаны.

Нельзя использовать FDISK для удаления non DOS Partition. Для загрузки ОС с non DOS Partition до удаления Primary DOS Partition измените ее статус на активный.

На винчестерском диске 1 нельзя уничтожить Primary DOS Partition, не уничтожив предварительно Extended DOS Partition. При попытке сделать это FDISK сообщит:

**Can not delete Primary DOS Partition on
drive 1 when Extended DOS Partition exists**

Уничтожение Extended DOS Partition

Если вам необходимо уничтожить Extended DOS Partition, на которой находятся несколько логических дисков, сначала удалите логические диски. Для удаления логического диска надо набрать на клавиатуре "3", выбирая в основном меню опцию Delete Logical DOS Drive(S), затем нажать "Enter". FDISK высветит на дисплее:

Delete Logical DOS Drive

Drv	Start	End	Size
D:	410	613	204
E:	614	818	205

Total partition space is 409 cylinders.

Warning! Data in the logical DOS drive will be lost. What drive do you wish to delete...? []

Press ESC to return to Fdisk Options

Надпись **Warning!** будет мигать.

Наберите букву, соответствующую диску, подлежащему удалению, и нажмите "Enter". FDISK высветит:

Are you sure...? [N]

Если логический диск содержит данные, не переписанные на другой носитель, то нажмите "Enter". Это не даст FDISK уничтожить логический диск. Если вы уверены, что диск необходимо уничтожить, наберите [Y] и "Enter".

После уничтожения логических дисков можно удалить Extended DOS Partition, выбрав в основном меню 3. Delete DOS Partition, а затем 2. Delete Extended DOS Partition. FDISK сообщит:

Delete Extended DOS Partition

Current Fixed Drive: 1

Partition	Status	Type	Start	End	Size
c: 1	A	PRI DOS	0	409	410
2		EXT DOS	410	818	409

Warning! Data in Extended DOS Partition will be lost. Do you wish to continue...? [N]

Press ESC to return to FDISK Options

Надпись **Warning!** будет мигать.

Для удаления Extended DOS Partition надо набрать в скобках [Y] и нажать "Enter".

Если вы передумали уничтожать Extended DOS Partition, нажмите "Enter" при ответе в скобках [N].

Уничтожение Primary DOS Partition

В меню "Delete DOS Partition" наберем на клавиатуре "1" и нажмем "Enter". Перед нами возникнет меню:

Delete Primary DOS Partition

Current Fixed drive: 1

Partition	Status	Type	Start	End	Size
c: 1	A	PRI DOS	0	409	410

**Warning! Data in Primary DOS
Partition will be lost. Do you wish
to continue...? [N]**

Press ESC to return to FDISK Options

Надпись **Warning!** будет мигать.

Для удаления Primary DOS Partition надо набрать "Y" и нажать "Enter".

Если вы передумали уничтожить Primary DOS Partition, нажмите "Enter" при ответе в скобках [N].

Как выбрать другой винчестерский диск?

Опция 5. **Select Next Fixed Disk Drive** появляется в основном меню FDISK только в том случае, если в вашей ПЭВМ есть несколько винчестерских дисков. При вызове этой опции FDISK изменяет текущий диск на следующий. Например, если текущий - **HARD DISK DRIVE 1** и вы выбрали опцию 5. **Select Next Fixed Disk Drive** основного меню, то FDISK перейдет на текущий диск 2.

Когда вы выберете следующий диск, FDISK снова высветит основное меню. В верхней строчке меню будет написано

Current Fixed Disk Drive: 2

В дальнейшем FDISK с винчестерским диском 2 работает так, как описано выше.

Отметим, что Partition, совместимые с другими ОС, создаются средствами этих систем.

Примечание. Данная работа была выполнена на ПЭВМ класса IBM PC/AT и винчестерском диске ST-251 фирмы Seagate.

УДК 681.3.06

Г.Г. Гнездилова

НАЦИОНАЛЬНЫЙ АЛФАВИТ И ПЕРСОНАЛЬНЫЙ КОМПЬЮТЕР

Один философ сказал, что различие языков - величайшее бедствие человечества. Сейчас можно смело утверждать, что величайшее бедствие пользователя персонального компьютера (ПК) - это различие алфавитов. Большие сложности ждут обладателя отечественного компьютера ЕС1840, пытающегося запустить популярный Лексикон, не столь широко распространенный Политекст или профессиональный MS WORD, и уж совсем непосильные тяготы ложатся на плечи любого, кто работает на компьютере западного производства, будь то профессиональный программист или неискушенный новичок. Буквы Г и Т вместо рамок, таинственная буква р, которую никак не удастся ввести с клавиатуры, темный прямоугольник вместо желанной буквы А могут довести до отчаяния при первом же запуске машины или нового программного продукта.

Попробуем проанализировать возникшую проблему и один из возможных подходов к ее решению.

Представление символов в персональном компьютере

Задавались ли вы когда-нибудь вопросом, как представляются в компьютере те символы (литеры, знаки), которые вы вводите с клавиатуры, видите на экране дисплея или на листе бумаги, выведенном с принтера? Если нет, посмотрите на табл. 1. Она содержит первые 128 символов из числа тех, с помощью которых вы можете обмениваться информацией с компьютером, и называется таблицей символов в коде ASCII (аббревиатура от American Standard for Information Interchange). Какие символы, или, точнее, какие группы символов входят в эту таблицу, рассмотрим чуть позже. Сейчас же для нас существенно, что каждый символ в таблице получает номер, который может быть вычислен как $16xj+i$ (где i и j - соответственно номера

строки и столбца, в которых расположен символ). Этот номер и представляет символ в компьютере. Именно код записывается в память компьютера (когда вводите с клавиатуры представляемый им символ), передается служебной программе для вывода на дисплей (когда хотите изобразить символ на экране) или посылается в принтер (когда хотите символ напечатать). Способ кодирования символов (табл. 1) - стандарт для современных компьютеров, поэтому обсудим его подробнее.

Таблица 1

Таблица символов в коде ASCII

	0	1	2	3	4	5	6	7
0	NUL	DLE		0	@	P	\	p
1	SON	DC1	!	1	A	Q	a	q
2	STX	DC2	"	2	B	R	b	r
3	ETX	DC3	#	3	C	S	c	s
4	EOT	DC4	\$	4	D	T	d	t
5	ENQ	NAK	%	5	E	U	e	u
6	ACK	SYN	&	6	F	V	f	v
7	BEL	ETB		7	G	W	g	w
8	BS	CAN	(	8	H	X	h	x
9	HT	EM	)	9	I	Y	i	y
10	LF	SUB	*	:	J	Z	j	z
11	VT	ESC	+	;	K	[	k	{
12	FF	FS	,	<	L	\	l	
13	CR	QS	-	=	M	]	m	}
14	SO	RS	.	>	N	.	n	-
15	SI	US	/	?	O	_	o	

Таблица содержит две основные группы символов (управляющие и печатаемые), различающиеся адресатом несомой информации.

Управляющие символы, как правило, используются для управления печатью или при передаче данных по линиям связи. Эти символы адресованы служебным программам, выполняющим вывод на принтер, экран или в коммуникационный канал. Например, вывод управляющих символов CR (возврат каретки) и LF (новая строка) на экран или печатающее устройство трактуется соответствующими программами как приказ подготовиться к выводу последующих символов с начала новой строки. Управляющие символы расположены в первых

двух столбцах таблицы и кодируются числами от 0 до 31 включительно. Изображения символов стандартом не фиксируются и могут различаться на разных компьютерах.

Печатаемые символы предназначены для записи текстовой и числовой информации. В эту группу входят буквы латинского алфавита, цифры, знаки препинания, арифметических операций и операций сравнения, скобки, а также некоторые другие, например @, # или &.

Табл. 1 не исчерпывает всех символов, доступных ПК: всего их 256, каждый кодируется числом от 0 до 255. Для хранения кода символа в памяти ПК отводится только один байт (восемь двоичных разрядов). Наибольшее десятичное число, которое может быть представлено таким количеством разрядов, равно 255 (11111111 в двоичной системе счисления). Именно поэтому вы можете использовать только 256 символов. В отличие от первых 128 состав и способ кодирования символов с кодами от 128 до 255 не стандартизованы и различаются на разных компьютерах. Как правило, к ним относятся буквы национальных алфавитов, математические знаки и символы псевдографики, т.е. символы для построения

Таблица 2

Расположение символов псевдографики в таблице символов IBM PC

	8	9	10	11	12	13	14	15
0					L	ll		
1				⌘	┐	┐		
2				⌘	┐	π		
3					┐	ll		
4				┐	—	┐		
5				┐	┐	F		
6				ll	┐	π		
7				π	ll	ll		
8				┐	ll	ll		
9				ll	π	┐		
10				ll	ll	ll		
11				π	π	■		
12				ll	ll	■		
13				ll	=	ll		
14				┐	ll	ll		
15				┐	┐	■		

несложных графических изображений (например, разграфки таблиц или штриховки прямоугольников). На ПК IBM PC XT/AT используются символы из табл. 2.

На отечественном компьютере ЕС1840 (41, 42, 45) используются символы из табл. 3. В этой таблице все русские буквы, кроме Ё и ё, занимают позиции с номерами $176_{10} \dots 239_{10}$ ($BO_{16} \dots EF_{16}$), где располагаются в алфавитном порядке. Буквы Ё и ё находятся соответственно в позициях 161_{10} ($A1_{16}$) и 241_{10} ($F1_{16}$).

Таблица 3

Расположение русских букв в
основной (ГОСТ) таблице символов

	8	9	10	11	12	13	14	15
0				А	Р	а	р	
1			Ё	Б	С	б	с	ё
2				В	Т	в	т	
3				Г	У	г	у	
4				Д	Ф	д	ф	
5				Е	Х	е	х	
6				Ж	Ц	ж	ц	
7				З	Ч	з	ч	
8				И	Ш	и	ш	
9				Й	Щ	й	щ	
10				К	Ъ	к	ъ	
11				Л	Ы	л	ы	
12				М	Ь	м	ь	
13				Н	Э	н	э	
14				О	Ю	о	ю	
15				П	Я	п	я	

Исходный принцип разработки табл. 3 - следование рекомендациям ISO: не занимать два первых столбца существенными для представления информации символами (поскольку подобно символам с кодами из диапазона $0 \dots 31$ они часто используются как управляющие и возможность их изображения ограничена). Русские буквы в табл. 3 занимают те столбцы, которые в табл. 2 отведены символам псевдографики. Именно по этой причине при запуске на ЕС1840 программ, разрабо-

танных для IBM PC, вместо рамок вы видите русские буквы й, к, л и др.

Для работы с русским алфавитом (кроме табл. 3) активно используется табл. 4. В отличие от табл. 3 табл. 4 совместима с табл. 2 по символам псевдографики.

Таблица 4

Альтернативная таблица символов

	8	9	10	11	12	13	14	15
0	А	Р	а				р	Ё
1	Б	С	б				с	ё
2	В	Т	в				т	
3	Г	У	г				у	
4	Д	Ф	д				ф	
5	Е	Х	е				х	
6	Ж	Ц	ж				ц	
7	З	Ч	з				ч	
8	И	Ш	и				ш	
9	Й	Щ	й				щ	
10	К	Ъ	к				ъ	
11	Л	Ы	л				ы	
12	М	Ь	м				ь	
13	Н	Э	н				э	
14	О	Ю	о				ю	
15	П	Я	п				я	

В ней полностью сохранен их набор и расположение. Все русские буквы в этой таблице, кроме букв Ё и ё, занимают позиции с номерами $128_{10} \dots 175_{10}$ ($80_{16} \dots AF_{16}$) и $224_{10} \dots 239_{10}$ ($E0_{16} \dots EF_{16}$), где располагаются в алфавитном порядке. Буквы Ё и ё находятся соответственно в позициях 240_{10} ($F0_{16}$) и 241_{10} ($F1_{16}$). При таком способе кодирования правильно изображаются разграфка таблиц, рамки окон и прочие элементы интерфейса современных программных систем, разработанных для IBM PC, но возникают некоторые трудности при программировании работы с русскими буквами. Так, чтобы определить, является ли заданный символ русской буквой, приходится отдельно проверять код символа на принадлежность двум разным диапазонам: $128 \dots 175$ (символ -

одна из букв от А до п), и 140...241 (символ - одна из букв от р до ё).

Способ кодирования русских букв, представленный в табл. 3, зафиксирован ГОСТом, сама таблица названа **основной**. Заметим, что на кодирование других символов, в том числе символов псевдографики, стандарт не распространяется - свободные позиции таблицы могут заполняться любыми нужными программисту символами.

Табл. 4 получила название **альтернативной** и была рекомендована авторами для обеспечения совместимости по символам псевдографики с программным обеспечением IBM PC. В табл. 3 последний столбец может "плавать" весь, кроме двух верхних позиций, которые также отданы русским буквам. Итак, мы знаем, как кодируются символы. А как они получаются на экране?

Изображение символов на экране дисплея

Шаблон символа

Изображение на экране дисплея образуется подсветкой отдельных областей. Минимальная область, которая может быть подсвечена аппаратурой, называется **элементом раstra**, **пикселем** (от английского слова pixel - picture element) или **точкой экрана** (мы будем придерживаться этого последнего наименования). В начальный момент все точки экрана окрашены в одинаковый цвет, например черный или синий. Такое состояние экрана ассоциируется с отсутствием на нем изображения.

Чтобы получить на экране изображение какой-либо буквы, нужно окрасить точки, которые составят ее контур, цветом, отличным от фонового. В памяти компьютера хранятся данные, кодирующие изображение всех 256 символов: каждому коду (числу от нуля до 255) ставится в соответствие совокупность чисел, задающих начертание символа с этим кодом. Для кодирования используется **шаблон** - прямоугольное поле, состоящее из определенного числа клеток (на рис. 1 показан шаблон буквы А). Каждая клетка шаблона в соответствии с начертанием символа закрашивается или остается пустой.

Количество клеток, составляющих шаблон, различается для разных шрифтов и на разных адаптерах. (Адаптер дисплея - это специальное устройство, обеспечивающее сопряжение дисплея и компьютера, а также изображение данных на экране.) Например, для цветного графического адаптера CGA IBM PC и для графического адаптера ECI840 шаблон имеет

размеры 8x8 клеток, для монохромного адаптера ЕС1840 - 8 точек по горизонтали и 14 точек по вертикали. Для работы с расширенным дисплейным адаптером EGA и адаптером Hercules Plus в разных режимах используются шаблоны двух размеров: 8x8 и 8x14 клеток, с адаптером VGA - три шаблона размерами 8x8, 8x14 и 8x16 клеток и т.д. Заметьте, что в шаблоне учитываются горизонтальные и вертикальные промежутки, отделяющие символы и строки символов друг от друга. Эти промежутки задаются пустыми строками и столбцами шаблона (на рис. 1 пусты два правых столбца и нижняя строка). Ясно, что чем "крупнее" шаблон, тем с большей точностью с его помощью можно задать начертание символа.

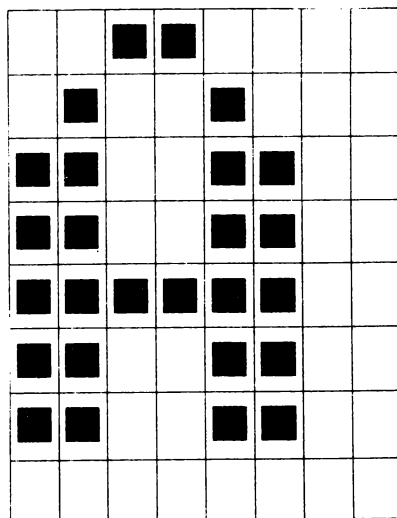


Рис.1 Шаблон буквы А

При выводе символа на экран его шаблон сопоставляется с прямоугольной областью экрана, в которой должно быть построено изображение символа: каждой клетке шаблона ставится в соответствие одна точка экрана. Соответствие может быть и другим, например, каждой клетке - область экрана размерами 2x2 точки, каждой клетке - две точки, соседние по горизонтали или по вертикали, и т.п. Точка, поставленная в соответствие закрашенной клетке, окрашивается цветом, отличным от фоновой, а точка, сопоставляемая с пустой клеткой, - цветом фона.

Шаблон кодируется последовательностью нулей и единиц (закрашенная клетка - единицей, пустая - нулем). Полученный результат называется растром символа. Зная растры всех символов, можно по заданному коду символа получить его изображение на экране. Рассмотрим, как хранятся в ПК растры символов и что происходит после того, как вы дали команду вывести на экран нужный символ (применительно к широко распространенному в нашей стране появившемуся одним из первых цветному графическому адаптеру CGA). За исключением некоторых деталей, рассуждения применимы и к более совершенным - профессиональным адаптерам, таким как EGA или VGA (краткое описание этих, а также некоторых других адаптеров даны в следующем разделе). Различия обусловлены дополнительными возможностями, например увеличением числа символов, выводимых на экран, или количества доступных цветов, поэтому переход к работе с такими адаптерами не вызовет серьезных затруднений.

Дисплей персонального компьютера (адаптер CGA)

Чтобы вывести символ на экран дисплея, нужно в специальную область оперативной памяти, называемую **дисплейной памятью**, или **видеопамятью**, записать специальную информацию. Дисплейная память имеет размер 16К и хранит данные об изображении, сформированном на экране. Буква К обозначает 1 Кбайт, в состав которого входит 1024 байта. Таким образом, 16К равно 16х1024, т. е. 16384 байтам. Дисплейная память при работе с адаптером CGA располагается по шестнадцатеричному адресу В8000. Необходимая информация формируется и заносится в дисплейную память специальными системными программами, которые начинают работать по команде вывода символа на экран, а затем считывается из нее аппаратурой и используется для построения нужного изображения. Дисплейная память считывается 50 раз в секунду, с такой же частотой обновляется изображение на экране. Состав информации, заносимой в дисплейную память, определяется режимом работы дисплея: символьным или графическим.

Для программиста символьные и графические режимы различаются теми минимальными объектами, которыми он может управлять: в **графическом режиме** - точкой экрана, в **символьном режиме** изображение строится из более крупных "строительных блоков" - символов. При работе в символьном режиме нельзя управлять отдельными точками экрана, участ-

вующими в построении изображения символа (например, нельзя задать свой цвет для каждой такой точки). Возможные режимы работы дисплея для адаптера CGA, а также их краткое описание даны в табл. 5.

Таблица 5

Режимы работы дисплея (адаптер CGA)

Номер режима	Краткое описание
0*	Символьный режим. Экран разбивается на 25 строк по 40 символов. Изображение черно-белое
1	Символьный режим. Экран разбивается на 25 строк по 40 символов. Могут использоваться 16 цветов из 16 возможных
2	Символьный режим. Экран разбивается на 25 строк по 80 символов. Изображение черно-белое
3	Символьный режим. Экран разбивается на 25 строк по 80 символов. Могут использоваться 16 цветов из 16 возможных
4	Графический режим. Экран разбивается на 200 строк по 320 точек в каждой. Могут использоваться 4 цвета из 16 возможных
5	Графический режим. Экран разбивается на 200 строк по 320 точек в каждой. Изображение черно-белое
6	Графический режим. Экран разбивается на 200 строк по 640 точек в каждой. Изображение черно-белое.

* В режимах 0, 2 и 5 цвет действительно подавляется, если к компьютеру подключен композитный монитор.

Нумерация режимов, приведенная в табл. 5, соответствует нумерации, принятой разработчиками базовой системы ввода-вывода - системной программы нижнего уровня. Нумерация режимов в табл. 5 может отличаться от нумерации, принятой в системе программирования, с которой вы работаете. Эта нумерация едина для всех видеоадаптеров. Если какой-либо режим не поддерживается адаптером, его таблица содержит скачок в нумерации (как, например, для адаптера EGA, см. табл. 6) или начинается ее не с нуля (компьютеры PCjr).

Таблица 6

Основные режимы работы дисплея (адаптер EGA)

Номер режима	Краткое описание
0...6	Совпадает с режимами работы адаптера CGA
7	Символьный режим. Экран разбивается на 25 строк по 80 символов. Изображение черно-белое (режим работы адаптера MDA)
13	Графический режим. Экран разбивается на 200 строк по 320 точек в каждой. Можно использовать 16 цветов из 16 возможных
14	Графический режим. Экран разбивается на 200 строк по 640 точек в каждой. Можно использовать 16 цветов из 16 возможных
15	Графический режим. Экран разбивается на 350 строк по 640 точек в каждой. Изображение черно-белое
16	Графический режим. Экран разбивается на 350 строк по 640 точек в каждой. Можно использовать 16 цветов из 16 возможных

В символьных режимах экран разбивается на 25 строк, в каждой из которых в зависимости от номера режима может быть изображено по 40 или 80 символов. Символы выводятся в фиксированные позиции экрана, которые называются **символьными позициями**, или **знакоместами**. В символьном ре-

жиме в дисплейной памяти хранятся сведения о всех символьных позициях экрана: номер цвета, который должна иметь позиция, и номер цвета, который должен иметь символ. Для хранения этих данных в дисплейной памяти каждой позиции отводятся два байта. В первом байте хранится код символа, выведенного в соответствующую позицию, разряды второго байта распределяются так:

		Второй байт		
М		Цвет фона	Я	Цвет символа
[7]		[6 5 4]	[3]	[2 1 0]

Биты 0...2 содержат базовый номер цвета символа. Таких цветов восемь: 0 - черный, 1 - синий, 2 - зеленый, 3 - голубой, 4 - красный, 5 - фиолетовый, 6 - коричневый, 7 - белый. Бит 3 - управляющий: если он содержит единицу, символ высвечивается с большей яркостью. Зрительно это проявляется как изменение цвета, например коричневого на желтый. Биты 4...6 содержат номер цвета, в который окрашивается позиция до вывода в нее символа, и номер создающего его фон. Для задания цвета фона символа используется та же нумерация цветов, что и для задания цвета символа. Бит 7, как и бит 3, - управляющий: наличие в нем единицы означает, что символ должен мерцать.

В графических режимах экран разбивается на 200 строк по 320 или по 640 точек в каждой. Слово "графический" не означает, что на экран нельзя выводить тексты. Как и в специальном режиме, символы выводятся обычными операторами вывода. При этом сохраняется та же разбивка экрана на символьные позиции: в режимах 4 и 5 их насчитывается 25 строк по 40 позиций, в режиме 6 - 25 строк по 80 позиций. Тем не менее, графический режим позволяет выполнять на экране различные геометрические построения или работать со "своими", разработанными программистом, буквами и цифрами, которые могут иметь иные, отличные от стандартного, размеры и начертания. В графическом режиме в дисплейной памяти хранятся номера цветов всех точек экрана. В режимах 4 и 5 каждой точке экрана отводятся два соседних бита. Поскольку с помощью двух двоичных разрядов можно записать только числа 0, 1, 2 и 3 (двоичные числа 00, 01, 10, 11), количество цветов, в которые может быть окрашено изображение в этом режиме, ограничено четырьмя. Номер 0 означает, что точка окрашена цветом фона, информация о котором хранится отдельно. Числа 1...3 - это номера цветов из двух доступных и не смешиваемых гамм, первая из них образована цветами: зе-

ленным (1), красным(2), коричневым (3), вторая - цветами;голубым (1), сиреневым (2), белым (3).

Каждый байт дисплейной памяти в графических режимах 4 или 5 хранит информацию о цветах четырех соседних точек. Так, первые 80 байт описывают цвета всех точек верхней строки экрана:

Номер точки экрана	0 1 2 3	4 5 6 7	316 317 318 319
Номер байта дисплейной памяти	Байт 0	Байт 1 ...	Байт 79

В режиме 6 на каждую точку экрана отводится только один бит, а каждый байт хранит информацию о восьми соседних точках экрана. Каждый бит может содержать только одно значение: 0 или 1, поэтому цветов всего два: 0 - черный, 1 - белый.

Обратите внимание на то, что, хотя в графических режимах и возможен вывод символов, информация о кодах выведенных символов в дисплейной памяти не содержится. В ней хранятся лишь номера цветов точек экрана, в том числе точек, составляющих контур символа.

Символ в текстовом режиме выводится так: по команде вывода символа в соответствующие два байта дисплейной памяти заносятся код символа и его цветовые атрибуты. При считывании этой информации адаптер дисплея по коду символа находит в принадлежащей ему области памяти (эта область называется **знакогенератором** дисплея) растр символа с заданным кодом. После этого точки символьной позиции, сопоставленные единичками растра, окрашиваются цветом символа, а остальные - цветом фона. Знакогенератор адаптера CGA не может быть изменен программно: чтобы сменить начертание символов, например для работы с русским алфавитом, нужно с помощью специального устройства (программатора) записать в знакогенератор новые коды - "перешить" знакогенератор. Другие адаптеры, например графический адаптер ECI840 или расширенный адаптер EGA, имеют **загружаемый знакогенератор**, т.е. позволяет программисту задавать новые растры программным путем.

В графическом режиме работа с растром предшествует записи в дисплейную память. По команде вывода символа с заданным кодом системными программами в определенной области памяти находится растр нужного символа и переписывается в дисплейную память. Поэтому изображение регенери-

руется только по содержимому дисплейной памяти без дополнительного обращения к знакогенератору дисплея. Растры символов с кодами 0...217, используемые для занесения в дисплейную память, хранятся в ПЗУ компьютера (там же, где и программы базовой системы ввода-вывода). Растры 128 символов должны быть сформированы прикладной программой, а содержащая их таблица записана по определенному адресу.

Адаптеры дисплея (краткая справка)

Из нескольких сотен адаптеров, предназначенных для компьютеров фирмы IBM, широко распространены всего пять: CGA, MDA, HGC, EGA и VGA.

Адаптер CGA (Color Graphic Adapter), о котором мы говорили до сих пор, - один из первых видеоадаптеров для персональных компьютеров (создан в 1981 г.) Почти одновременно появился адаптер MDA (Monochrome Display Adapter), который в отличие от CGA поддерживает только **монохромный**, т. е. черно-белый, алфавитно-цифровой режим работы (в нумерации режимов имеет номер 7).

Немного позднее (в 1982 г.) был создан адаптер HGC (Hercules Graphics Card), а затем (в 1986 г.) - адаптер HGC+ (Hercules Graphics Card Plus). Оба адаптера быстро завоевали признание в области монохромной графики. По сравнению с CGA и MDA эти адаптеры позволяют использовать шрифты более высокого качества; размеры шаблона - 8x14 точек. В отличие от HGC знакогенератор адаптера HGC+ - загружаемый.

Адаптер EGA (Enhanced Graphics Adapter) появился в 1985 г. Он позволяет работать в текстовом и графическом режимах. Максимальное разрешение в графическом режиме - 640x350 точек, можно использовать 16 цветов из 64 возможных (см. табл. 6).

Основные режимы работы дисплея (адаптер EGA)

Номер режима	Краткое описание
0...6	Совпадает с режимами работы адаптера CGA
7	Символьный режим. Экран разбивается на 25 строк по 80 символов. Изображение черно-белое (режим работы адаптера MDA)

- | | |
|----|---|
| 13 | Графический режим. Экран разбивается на 200 строк по 320 точек в каждой. Можно использовать 16 цветов из 16 возможных |
| 14 | Графический режим. Экран разбивается на 200 строк по 640 точек в каждой. Можно использовать 16 цветов из 16 возможных |
| 15 | Графический режим. Экран разбивается на 350 строк по 640 точек в каждой. Изображение черно-белое |
| 16 | Графический режим. Экран разбивается на 350 строк по 640 точек в каждой. Можно использовать 16 цветов из 16 возможных |
-

Новый графический адаптер VGA (Video Graphics Array) появился в 1988 г. на компьютере PS/2. Как и EGA, адаптер VGA поддерживает алфавитно-цифровые и графические режимы. Максимальное разрешение в графическом режиме составляет 640x480 точек. Помимо основных режимов (табл. 6) возможны еще режимы 17...19. Поддерживается одновременная работа с 256 цветами из 262144. Такое большое количество цветов обусловлено тем, что адаптер использует не цифровые, а аналоговые сигналы управления цветом. Адаптер VGA фирмы IBM совместим только с аналоговыми мониторами (как цветными, так и монохромными). Однако многие совместимые с ним адаптеры других фирм способны работать с цифровыми мониторами; число отображаемых цветов при этом - до 64.

Кроме перечисленных используется еще ряд адаптеров, например MCGA (Multi-Color Graphics Array), младших моделей PS/2, Hercules InColor Card, Hercules Color Card и др. Но даже адаптеры одного вида часто отличаются друг от друга. Например, адаптер EGA с внутренней оперативной памятью в 256 Кбайт работает, как описано в табл. 6. Тот же адаптер, оснащенный 64 Кбайт памяти, оперирует только с четырьмя цветами. Максимальное разрешение современных модифицированных адаптеров - 640x480 точек.

Ввод с клавиатуры

Очевидно, что буквы национального алфавита нужно уметь не только выводить на экран, но и вводить с клавиатуры. Клавиатура - это главное средство ввода информации в компьютер. В основном с помощью клавиатуры вы задаете значения исход-

ных данных, вводите нужные тексты, выбираете пункты меню, управляете движущимися по экрану объектами и т. д. Клавиатуры отечественных компьютеров позволяют вводить и русские, и латинские буквы. Из режима ввода латинских букв в режим ввода русских букв и обратно клавиатура переключается специальной клавишей РУСС.

На клавиатурах зарубежного производства режим ввода букв национальных алфавитов не предусмотрен. Единственный способ ввести с такой клавиатуры символ с кодом, большим 127, - это набрать нужный код на дополнительной цифровой клавиатуре при одновременном нажатии клавиши Alt. Ясно, что такой способ ввода неприемлем. Обеспечить возможность удобной работы с русским алфавитом можно только ценой вмешательства в работу компьютера, почти на самом низком уровне - на уровне базовой системы ввода-вывода.

Базовая система ввода-вывода образована несколькими программами, составляющими промежуточные звенья между операционной системой (ОС) и внешними устройствами. Эти программы скрывают от ОС технические подробности функционирования конкретных внешних устройств и включаются в работу по специальным запросам, выдаваемым аппаратурой или программами. Временное прекращение текущей работы компьютера с целью обслуживания поступившего запроса называют прерыванием, программу, обслуживающую запрос, - обработчиком прерывания.

Для обработки данных, поступающих от клавиатуры, используются прерывания двух видов: первое (в системе прерываний ПК оно имеет номер 9) обеспечивает прием с клавиатуры и предварительную обработку сведений о нажатых клавишах, второе (имеет шестнадцатеричный номер 16) предоставляет полученные данные в распоряжение прикладной программы или ОС; 9-е прерывание автоматически выполняется по запросу, поступившему от клавиатуры, 16-е - по команде прикладной программы или ОС. Для "встраивания" русских букв мы попробуем вмешаться в работу 9-го прерывания. Заметим, что делать это позволительно только квалифицированному системному программисту.

Если вы уже работали на IBM PC, то, вероятно, использовали клавишу Shift, которая меняет регистр клавиши, нажимаемой одновременно с ней. Именно с помощью этой клавиши вы можете начать с прописной буквы предложение, имя или название.

Если весь текст должен набираться прописными буквами, можно воспользоваться клавишей CapsLock, в результате ее нажатия буквы, набираемые без клавиши Shift, будут пропис-

ными, для ввода строчных букв потребуется клавиша Shift. Повторное нажатие клавиши CapsLock восстанавливает начальное состояние клавиатуры: буквы, набираемые на нижнем регистре, вновь становятся строчными, буквы, набираемые на верхнем, - прописными. Попробуем изменить назначение этой клавиши. Попытаемся сделать так, чтобы ее последовательные нажатия переключали клавиатуру из режима ввода латинских букв в режим ввода русских букв и обратно. Разумеется, мы потеряем полезную клавишу, но зато получим возможность работы с русским алфавитом. А теперь небольшое отступление, в ходе которого станет ясно, как должна быть устроена программа, обеспечивающая упомянутую возможность.

Стандартная клавиатура IBM PC XT

Клавиатура IBM PC XT и совместимых с ней ПК зарубежного производства имеет символьные клавиши с буквами, цифрами, знаками пунктуации, арифметических действий и некоторыми другими, а также управляющие клавиши. К числу управляющих относятся функциональные клавиши (F1...F10), клавиши управления курсором (Up, Down, Left, Right, Home, End, PgUp, PgDn), клавиши редактирования (Backspace, Ins, Del, Tab), регистровые и другие клавиши, изменяющие смысл нажимаемых одновременно с ними (Shift, Ctrl, Alt, CapsLock, NumLock, ScrollLock), а также еще три специальные клавиши (Esc, Enter, PrtSc). Каждая клавиша имеет номер - число из диапазона 1...83. Всякий раз, когда вы нажимаете или отпускаете клавишу, клавиатура вырабатывает однобайтовый код (он называется скэн-кодом), идентифицирующий ваше действие. При нажатии клавиши скэн-код представляет собой номер клавиши, при ее отжатии - число, на 128 большее (старший бит установлен в единичное состояние), например, при нажатии клавиши Z скэн-код равен 44, при отжатии - 172(44+128).

На рис. 2 показаны клавиши, которые будут использоваться для ввода русских букв, и их скэн-коды. Таким образом, никакой информации о вводимых символах в компьютер от клавиатуры не поступает, только сведения о том, какие клавиши и в каком порядке были нажаты. Более содержательные данные, которые могут использоваться прикладной программой, формируются базовой системой ввода-вывода и предоставляются прикладной программе.

Q/Й	W/Ц	E/У	R/К	T/Е	Y/Н	U/Г	I/Ш	O/Щ	P/З	{/Х	
16	17	18	19	20	21	22	23	24	25	26	
A/Ф	S/Ы	D/В	F/А	G/П	H/Р	J/О	K/Л	L/Д	:/Ж	"/Э	~/Ъ
30	31	32	33	34	35	36	37	38	39	40	41
Z/Я	X/Ч	C/С	V/М	B/И	N/Т	M/Ь	</Б	>/Ю			
44	45	46	47	48	49	50	51	52			

Рис. 2. Буквенные клавиши и их скэн-коды

Механизм взаимодействия клавиатуры и базовой системы ввода-вывода состоит в следующем. Сформированный клавиатурой скэн-код записывается в определенный порт (портом называют байт дополнительной памяти ПК, предназначенный для связи с внешним устройством), одновременно выдается запрос на прерывание - поднимается флажок, сообщающий о появлении очередного скэн-кода. Работа, выполняемая ПК, автоматически приостанавливается, и начинает обрабатываться поступивший запрос (выполняется 9-е прерывание). Программа-обработчик 9-го прерывания читает порт клавиатуры, а затем преобразует однобайтовый код в двухбайтовый: младший (основной) байт построенного ею кода обычно содержит ASCII код введенного символа, старший байт (вспомогательный) - номер соответствующей клавиши. Полученный двухбайтовый код записывается в специальную область оперативной памяти компьютера (буфер клавиатуры), откуда затем может читаться обработчиком 16-го прерывания, выполняемого по запросу от прикладной программы или ОС.

Содержимое старшего байта игнорируется, если прикладной программе безразлично, с помощью какой клавиши введен тот или иной символ. В других случаях эта информация оказывается полезной, например, если на клавиатуре есть две клавиши с одним и тем же символом (как при знаке +, который можно ввести и с помощью правой верхней клавиши знаковой клавиатуры и с помощью правой нижней управляющей клавиши; дублируются и другие знаки, например все цифры, звездочка, точка) и для прикладной программы нажатия этих клавиш должны вызывать разные действия. Например, в некоторых системах знак +, введенный символьной клавишей, вставляется в редактируемый текст, а нажатие управляющей клавиши "серый плюс" трактуется как приказ запустить программу, на имя которой указывает курсор.

Если символ вводится с помощью дополнительной цифровой клавиатуры, вспомогательный байт содержит нуль. Это

также позволяет прикладной программе при необходимости установить, каким способом был введен символ. Для большинства управляющих клавиш и их комбинаций, а также комбинаций знаковых клавиш с клавишей Alt и некоторых других основной байт содержит нуль, а вспомогательный - число, позволяющее идентифицировать выполненное нажатие. Ниже перечислены наиболее часто используемые в прикладных программах клавиши и комбинации клавиш, нажатие которых транслируется в двухбайтовый код с нулевым основным байтом:

Функциональные клавиши

F1	59	Shift-F1	84	Ctrl-F1	94	Alt-F1	104
F2	60	Shift-F2	85	Ctrl-F2	95	Alt-F2	105
F3	61	Shift-F3	86	Ctrl-F3	96	Alt-F3	106
F4	62	Shift-F4	87	Ctrl-F4	97	Alt-F4	107
F5	63	Shift-F5	88	Ctrl-F5	98	Alt-F5	108
F6	64	Shift-F6	89	Ctrl-F6	99	Alt-F6	109
F7	65	Shift-F7	90	Ctrl-F7	100	Alt-F7	110
F8	66	Shift-F8	91	Ctrl-F8	101	Alt-F8	111
F9	67	Shift-F9	92	Ctrl-F9	102	Alt-F9	112
F10	68	Shift-F10	93	Ctrl-F10	103	Alt-F10	113

Клавиши управления курсором

Home	71	PgUp	73	Right	77	Up	72
End	79	PgDn	81	Left	75	Down	80

Клавиши редактирования

Ins	82	Del	83
-----	----	-----	----

Преобразование, выполняемое обработчиком 9-го прерывания, достаточно сложно и поэтому требует детально учитывать сделанную последовательность нажатий. Например, нажатие клавиши C транслируется в код, соответствующий строчной букве c, нажатие этой же клавиши в комбинации с клавишей Shift - в код, соответствующий прописной букве C, а в комбинации с клавишей Ctrl - в код, трактуемый ОС как сигнал на прекращение текущей работы. Заметим, что имеются клавиши, которым никакой код в соответствие не ставится. Это клавиши, модифицирующие коды других клавиш: LeftShift, RightShift, Ins, NumLock, StrollLock, Alt, Ctrl. К их числу относится и клавиша CapsLock. Факт нажатия этих клавиш отображается в специальном байте (первый байт ста-

туса клавиатуры по шестнадцатеричному адресу 0:417), разряды которого распределены между упомянутыми клавишами следующим образом:

7	6	5	4	3	2	1	0
Ins	CapsLock	NumLock	ScrollLock	Alt	Ctrl	LeftShift	RightShift

Биты, отведенные клавишам Alt, Ctrl, LeftShift и RightShift, получают единичное значение, когда вы нажимаете соответствующую клавишу, и сбрасываются в нуль, когда вы ее отпускаете. Каждое нажатие клавиши Ins, CapsLock, NumLock или ScrollLock поочередно переключает бит, отведенный клавише, из нулевого состояния в единичное и из единичного в нулевое. Когда базовая система получает скэн-код нажатой клавиши, она анализирует байт статуса клавиатуры и преобразует исходный скэн-код в соответствии с его состоянием.

Алгоритм ввода русских букв

Вернемся, наконец, к задаче ввода русских букв. Мы условились, что попытаемся использовать клавишу CapsLock для переключения клавиатуры из режима ввода латинских букв в режим ввода русских букв и обратно. Для этого дополним стандартную программу обработки 9-го прерывания анализом бита состояния клавиши CapsLock. Для определенности предположим, что в некоторый момент, назовем его начальным, этот бит имеет нулевое значение, что соответствует работе клавиатуры в режиме ввода латинских букв. (Чтобы это предположение было верным, нужный бит в начале работы программы можно сбросить в нулевое состояние.) Установка бита в единичное состояние стандартным обработчиком 9-го прерывания означает, что клавиатура была переведена в режим ввода русских букв нажатием клавиши CapsLock, последующий сброс в нулевое состояние - что клавиатура была возвращена этой клавишей в режим ввода латинских букв. Если клавиатура находится в режиме ввода латинских букв, никакие действия, кроме стандартных, выполняться не должны.

Если клавиатура находится в режиме ввода русских букв, код, полученный в результате стандартной обработки, может потребовать модификации. Изменение последнего двухбайтового кода, записанного в буфер клавиатуры, необходимо, когда этот код соответствует нажатию буквенной клавиши. При последнем условии ни основной, ни вспомогательный байты не

равны нулю (неравенство нулю основного байта означает, что был введен символ, неравенство нулю вспомогательного байта - что символ не был введен с помощью дополнительной цифровой клавиатуры) и вспомогательный байт (номер клавиши) содержит число, принадлежащее одному из диапазонов: 16...26 (верхний ряд буквенных клавиш, см. рис. 2), 30...41 (средний ряд буквенных клавиш), 44...51 (нижний ряд буквенных клавиш). Здесь мы не учитываем клавиши верхнего цифрового ряда, на отечественных пишущих машинках некоторые из них часто используются для ввода знаков “;”, и.

Пусть, например, основной байт двухбайтового кода, занесенного в буфер клавиатуры, содержит число 105 (код буквы *i*), а вспомогательный байт - число 23 (номер клавиши *I*). Это означает, что была нажата клавиша *I*, которая в режиме ввода русских букв предназначена для ввода буквы *ш* или *Ш* в зависимости от регистра, на котором эта клавиша была нажата). Таким образом, если бит состояния клавиши *LeftShift* или *RightShift* имеет единичное значение (клавиша *I* нажата на верхнем регистре), основной байт кода должен быть изменен на значение 152 (код прописной буквы *Ш*). В противном случае (клавиша *I* нажата на нижнем регистре) основной байт кода должен быть изменен на число 232 (код строчной буквы *ш*). Конкретное значение, которое должно быть получено в результате преобразования исходного значения основного байта, определяется клавиатурной раскладкой (т.е. соответствием между буквами, а в более общем случае между символами и клавишами) и выбранным способом кодирования. Клавиатурная раскладка для каждой клавиши задает букву, которая должна быть введена при нажатии этой клавиши (в нашем примере клавише *I* ставили в соответствие букву *ш* или *Ш*, а таблица кодировки - код введенной буквы (*Ш*-152, *ш*-232). В приведенном примере использована раскладка *ЙЦУКЕНГ*, стандартная для отечественных пишущих машинок, и альтернативная таблица символов.

О букве р

Большинство владельцев *IBM AT* уже, вероятно, столкнулись с проблемой буквы *р*, которую никак не удастся ввести. Связано это с тем, что основной байт кода, формируемого обработчиком 9-го прерывания для дополнительных управляющих клавиш расширенной клавиатуры, например функциональных клавиш *F11* и *F12*, устанавливается равным не нулю, как при

работе со стандартной клавиатурой, а 224 (шестнадцатеричное число E0). Делается это для того, чтобы прикладная программа могла разобрать, какая именно клавиша была нажата. Если вы посмотрите на таблицы кодировки, предназначенные для работы с русскими буквами, то убедитесь в том, что в обеих таблицах, и в основной, и в альтернативной, именно это число - код строчной русской буквы р. В результате букве р оказывается мощное сопротивление на всех уровнях работы с клавиатурой: и на уровне базовой системы ввода-вывода, и на уровне ОС, и на уровне прикладных пакетов.

Итак, если используется рассматривавшийся алгоритм ввода русских букв, ввод буквы р на уровне 9-го прерывания означает запись в буфер клавиатуры кода, основной байт которого содержит значение, равное E0, а старший - число 35 (номер клавиши, которая предназначена для ввода этой буквы). Чтобы прочитать буфер клавиатуры, прикладная программа должна обратиться к 16-му прерыванию через ОС или минуя ее. Обработчик 16-го прерывания, как и большинство других таких программ, не есть нечто неделимое - это совокупность взаимосвязанных подпрограмм. Каждая из них обеспечивает выполнение некоторой частной операции. На IBM PC XT в его состав входят подпрограммы, позволяющие прочитать из буфера клавиатуры очередной двухбайтовый код (имеет номер 0); определить, пуст ли буфер клавиатуры или в нем содержится какая-либо информация (имеет номер 1), и вернуть байт состояния клавиатуры (имеет номер 2). На IBM AT эти подпрограммы дополнены подпрограммой с шестнадцатеричным номером 10. Именно она обеспечивает чтение буфера клавиатуры с сохранением всей информации, специфичной для расширенной клавиатуры.

Подпрограмма с номером 0, сохраненная с целью обеспечения совместимости ПО, предназначена для работы с расширенной клавиатурой, как со стандартной. Нажатия дополнительных клавиш расширенной клавиатуры "подменяются" ею нажатиями управляющих клавиш или комбинаций клавиш стандартной клавиатуры. Делается это очень просто. Получив из буфера клавиатуры двухбайтовый код, основной байт которого имеет значение, равное E0, подпрограмма заменяет это значение на нуль (так как кодируется нажатие управляющих клавиш стандартной клавиатуры). Код 35 224, поступивший в буфер клавиатуры в результате ввода буквы р, заменяется ею на код 35 0, соответствующий комбинации клавиш Alt-35 стандартной клавиатуры (буква р пропадает).

Ситуацию иногда можно исправить, заменив в буфере клавиатуры код 35 224 на код 0 224. Если вы помните, такой

код должен быть получен при вводе буквы р с помощью дополнительной цифровой клавиатуры и клавиши Alt. Однако это поможет только в том случае, если подпрограмма с номером 0 написана "аккуратно", т. е. если замена кода 224 на 0 не выполняется, если вспомогательный байт равен нулю. К сожалению, это не всегда так, но ухищрения, которые приходится предпринимать программисту в таком случае, уже совсем не место в этой статье.

Заметим, что другие подобного рода сложности приходится преодолевать букве р и при прохождении через ОС. Но даже дойдя в конце концов до прикладного пакета, буква р может исчезнуть в нем. Это произойдет потому, что пакет, предназначенный для работы с расширенной клавиатурой, получив код, основной байт которого равен 224, может истолковать его как соответствующий управляющей клавише и проигнорировать, если обработка этой клавиши не предусмотрена (как например, широко распространенный NortonCommander), или выполнить действия, не соответствующие вашим намерениям. Единственная возможность преодолеть сопротивление букве р прикладного пакета - изменить его конфигурацию (если пакет позволяет это сделать), указав, что он должен работать не с расширенной, а со стандартной клавиатурой.

Вывод на принтер

Принтеры матричные, ромашковые, лазерные и другие

Среди периферийных устройств ПК принтеры представлены наибольшим количеством отличающихся друг от друга марок. Наиболее распространены матричные принтеры, входящие в комплект большинства компьютеров. Матричными называют принтеры, снабженные подвижной головкой с ударными стержнями - иглками. Управляя ударами стержней по бумаге, можно сформировать из отдельных точек узор, складывающийся в очертания нужного символа, при этом качество печати зависит от количества игловок в печатающей головке, чаще всего их 9 или 24.

В числе 9-игольчатых - такие как Epson FX-80, Epson FX-85, Epson LX-80, IBM Proprinter, RAVI 8010M, Juki 5500. В 9-игольчатых принтерах печатающие стержни расположены в головке по вертикали с шагом 1/72 дюйма (примерно 1/3 мм). Именно таким получается и расстояние между точками, составляющими символ, оно хорошо различается глазом, поз-

тому символ воспринимается не сплошным, как на пишущей машинке, а состоящим из отдельных точек. Такой текст вполне пригоден для черновиков, листингов программ и т. п., но не отвечает требованиям качественной печати. Расстояние между точками можно уменьшить до $1/216$ дюйма (примерно $1/9$ мм) по вертикали и $1/240$ дюйма ($1/10$ мм) по горизонтали. Однако для этого головка должна, во-первых, медленнее перемещаться, а во-вторых, печатать строку в три прохода со смещением бумаги на $1/216$ дюйма между проходами. В результате скорость снижается примерно в пять раз.

Наиболее распространены 24-игольчатые принтеры - Epson LQ, Epson LQ-500, Proprinter X24/XL24, NEC Pinwriter P6/P7. В этих принтерах стержни расположены в головке в два вертикальных ряда (по 12 иглол в ряд) с шагом $1/90$ дюйма (менее $1/3$ мм). Один ряд у них сдвинут относительно другого по вертикали на $1/180$ дюйма - расстояние между получаемыми при печати точками. Поэтому печать в один проход на 24-игольчатых принтерах примерно такого же качества, как на 9-игольчатых при печати в три прохода. Специальный режим высококачественной печати позволяет получить текст, практически не отличимый от полученного на пишущей машинке, однако скорость печати в этом режиме падает примерно в три раза.

Ромашковые принтеры, как и матричные, относятся к принтерам ударного действия, однако в отличие от матричных печатают с высоким качеством весь символ целиком (как и пишущая машинка), но не могут печатать картинки и не позволяют использовать одновременно несколько шрифтов. Для смены шрифта приходится заменять "ромашку" - плоское колесо с гибкими лепестками, на которых отштампованы выпуклые буквы.

Помимо принтеров ударного действия существуют принтеры безударные: лазерные, струйный и термографические. В лазерных принтерах электризуются отдельные участки вращающегося вала, из-за чего к ним прилипают частицы красящего порошка. Затем вал прокатывается по листу бумаги, а перешедший на бумагу порошок закрепляется на ней под воздействием тепла. Лазерные принтеры обеспечивают высокое качество печати и широко применяются на Западе (в частности, в настольных издательских системах для подготовки малотиражных изданий). Они обладают высоким быстродействием (чаще всего восемь страниц в минуту) и практически бесшумны, но стоят примерно столько же, сколько и ПК.

В струйных принтерах печать выполняется струей мельчайших капель чернил, в термографических - нагревом спе-

циальной чувствительной к теплу бумаги либо одноразовой ленты, покрытой пастообразной краской, которая плавится и переносится на бумагу. И струйные, и термографические принтеры сравнительно мало распространены, в частности потому, что требуют специальных чернил, бумаги или ленты, выпускаемых в небольших количествах специально для принтера каждой марки.

Вывод символов на матричном принтере

Задача поддержки национального алфавита на матричных принтерах разных марок решается средствами, близкими к тем, которые используются при работе с дисплеем (даже терминология одна и та же: шрифт, знакогенератор, символьный или графический режим и т. д.), и все же имеет огромное количество мелких различий.

Как и дисплей, принтер настроен на работу с определенным шрифтом, описание которого хранится в его ПЗУ. Такой шрифт называется **встроенным**. Встроенные шрифты большинства принтеров не имеют русских букв, поэтому, как и при работе с дисплеями, их приходится заменять. Способ установки нового шрифта на принтер зависит от того, позволяет ли принтер задавать новый шрифт программным путем - **загружать шрифт**, или шрифт жестко фиксирован и изменить его с помощью программы нельзя. В этом случае приходится поступать, как при работе с адаптером CGA: изменять знакогенератор с помощью программатора.

Задача формирования принтерного шрифта по сути повторяет задачу формирования шрифта дисплейного. Различаются лишь размеры шаблона и способы его кодирования. Размеры шаблона определяются прежде всего количеством игловок, собранных в печатающей головке принтера: как правило, их 9 или 24. У 9-игольчатых принтеров шаблон обычно имеет размеры 11 клеток по горизонтали на 8 клеток по вертикали, у 24-игольчатых принтеров - зависит от шрифта. Например, шаблон шрифта для черновой печати может иметь размеры 10 или 12 точек по горизонтали на 24 точки по вертикали, а шаблон шрифта для качественной печати - 30, 36 или 42 точки по горизонтали на 24 точки по вертикали. На рис. 3 показан шаблон буквы R для 9-игольчатого принтера и соответствующие ему коды. Из рисунка и поясняющей его таблички видно, что в отличие от дисплейного шаблона принтерный шаблон кодируется по столбцам:

Код шаблона:		
Номер столбца	Двоичный код	Шестнадцатеричный код
0	11111110	FE ₁₆
1	00000000	0 ₁₆
2	10010000	90 ₁₆
3	00001000	08 ₁₆
4	10010100	94 ₁₆
5	01100010	62 ₁₆
6	0	0 ₁₆
.....		
10	0	0 ₁₆

0	1	2	3	4	5	6	7	8	9	10
■		■		■						
■					■					
■					■					
■		■		■						
■			■							
■				■						
■					■					
FE16		9016		9416		016		016		016
	016		0816		6216		016		016	

Рис. 3. Кодирование шаблона буквы R

Сформированный код изображения загружается в **знакогенератор** принтера. Запись выполняется специальной командой. В нем указывается, изображение какого символа ею перепределяется. Так, если мы захотим, чтобы начертание символа с кодом 224 было задано шаблоном, показанным на рис. 3, потребуется сформировать следующую последовательность

(для ее записи воспользуемся нотацией языка Бейсик, в частности выражением CHR\$(N), где N - некоторое число, обозначение символа с кодом N):

```
CHR$(27); "&"; CHR$(0); CHR$(224);
CHR$(224);
CHR$(&H87);CHR$(&HFE);CHR$(0);CHR$(&H
90);CHR$(&H08);CHR$(&H94);CHR$(&H62);C
HR$(0);CHR$(0);CHR$(0); CHR$(0);CHR$(0)
```

В этой последовательности три первых символа - CHR\$(27), & и CHR\$(0) - признаки начала команды определения нового начертания символа. Число 224 - код переопределяемого символа. Оно записано дважды, поскольку команда задает начертание только одного символа. Если бы переопределялся не один, а несколько символов, вместо CHR\$(224); CHR\$(224) в команде было бы записано CHR\$(M); CHR\$(N), где M и N - числа, задающие диапазон кодов соответствующих символов.

Шестнадцатеричное число 87 задает дополнительную управляющую информацию, которая используется при печати символа. Старший двоичный разряд числа (рис. 4) указывает, нужно ли смещать символ на одну точку вниз. Именно так печатаются буквы с нижними (подстрочными) элементами: р, у, ц, щ и пр. Остальные семь разрядов задают область шаблона, используемую в режиме пропорциональной печати, т. е. когда печатается не весь шаблон целиком, а только некоторая его область; в нее включаются столбцы, "занятые" буквой, и столбцы, образующие промежутки между буквами. Ширина области индивидуальна для каждого символа, например, для буквы р она составляет семь столбцов, а для буквы ш - все 11 столбцов. Двоичные разряды с 7-го по 5-й задают левый столбец области, оставшиеся четыре разряда (с 4-го по 0-й) - ее правый столбец. Например, число 87₁₆, записываемое в двоичной системе счисления в виде 1 000 0111, означает, что символ нужно опускать вниз (старший бит равен нулю) и что при пропорциональной печати выводится область шаблона с 0-го (000) по 7-й (0111) столбец включительно.

Работа с принтером требует знания используемой им системы команд, которая, как правило, меняется от одной марки принтера к другой. Широко распространены две системы. Одна из них определяется стандартом фирмы Epson (стандарт ESC/P) - именно в этом стандарте нами была записана команда определения начертания буквы р, другая - стандартом фирмы IBM. Стандартный набор команд позволяет выполнить

такие действия, как инициализация принтера, загрузка нового шрифта, выбор шрифта и его модификация (например, жирный, подчеркнутый или курсив), переход на новую страницу

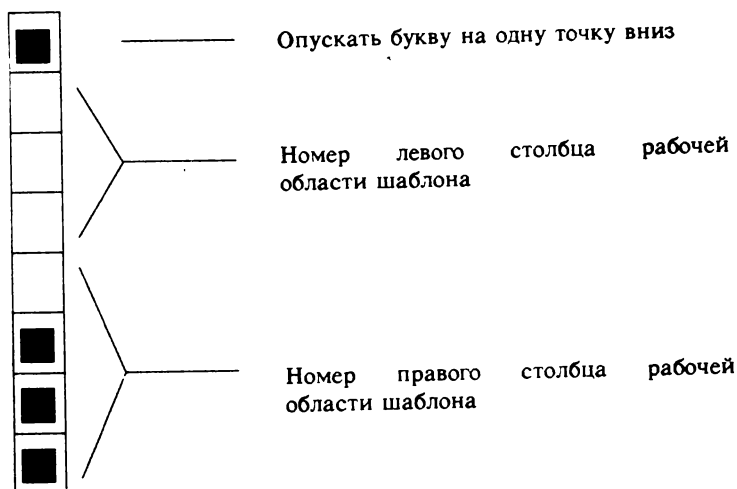


Рис. 4. Управляющая информация для печати буквы р

и пр. Большинство команд обоих стандартов записывается в виде ESC-последовательностей, т.е. последовательностей символов, начинающихся с управляющего символа ESC, например:

CHR\$(27); "@“ - инициализация принтера;

CHR\$(27); "%“; N - выбор шрифта с номером N, встроенного или загруженного;

CHR\$(27); "4“ - печать курсивом.

Нарисовав шаблоны всех букв нужного алфавита, закодировав их и загрузив в принтер полученные коды (о последнем действии чаще говорят как о загрузке символа или о загрузке шрифта, если речь идет о всех символах), мы получим возможность печати в символьном режиме, когда для вывода

символа достаточно указать один его код. При печати в **графическом режиме** программа должна рисовать символ каждый раз заново, самостоятельно выводя его шаблон столбец за столбцом. Такой способ более замедлен, но иногда может обеспечить нужное качество печати. Так, при работе с 9-игольчатыми принтерами часто требуется печатать буквы размером не 12x8, а 12x16 точек, т. е. в два раза более высокими. Единственный способ добиться этого - печатать каждую строку текста дважды: сначала "верхушки" символов, потом их оставшуюся часть. Сделать это можно, только используя графический режим печати.

Национальный алфавит и программное обеспечение

Мы уже говорили, что в результате переопределения второй половины таблицы символов может нарушиться обращенный к пользователю фасад прикладных программных систем. Чаше всего это происходит, когда символы псевдографики заменяются в стандартной таблице какими-либо другими символами, в частности буквами национального алфавита. Это несоответствие преодолеть нетрудно. Достаточно заменить используемую таблицу символов совместимой по символам псевдографики с таблицей IBM PC (табл. 3), однако символы псевдографики - это лишь часть проблемы.

Общая проблема заключается в том, что профессиональные пакеты, разработанные для латинского алфавита (такие, как например, система подготовки текстов MS WORD, интегрированная система Framework и др.), для работы с другими алфавитами не предназначены: эти системы считают буквами только символы с кодами от 65 до 90 и от 97 до 122. Что значит "считают буквами", ясно из следующего примера. MS WORD предоставляет возможность приписать любому фрагменту текста, например заголовку, специальный признак. В результате, на каком бы регистре ни был набран фрагмент, он всегда будет показываться прописными буквами, если буквы латинские, ведь для каждой буквы фрагмента нужно проверить, является ли она строчной, и, если это так, преобразовать ее в прописную, т.е. отнять от кода буквы 32. Именно на такое расстояние в стандартной ASCII-таблице прописные буквы отстоят от строчных. Проверка на принадлежность к строчным буквам выполняется по диапазону кодов - от 97 до 122. Коды русских букв, разумеется, в этот диапазон не входят. Кроме того, расстояния между прописными и строчными вариантами не одинаковы для всех букв.

Beta Plus - комплекс поддержки национального алфавита

Приведенные рассуждения показывают, что задача поддержки национального алфавита нетривиальна, и поэтому вряд ли должна решаться разработчиками каждой отдельной прикладной системы. **Комплекс Beta Plus**, разработанный в совместном советско-американском предприятии ПараГраф отделом А.А. Чижова, призван существенно улучшить состояние дел в этой области. **Beta Plus** позволяет использовать на ПК в дополнение к встроенному латинскому любой другой алфавит: буквы нужного алфавита могут быть введены с клавиатуры, изображены на экране дисплея и выведены на принтер. Национальный алфавит поддерживается на различных конфигурациях ПК типа IBM PC XT/AT/PS-2 в среде MS DOS.

Настройка **Beta Plus** на конкретные аппаратные средства ПК выполняется выбором и включением в **рабочую конфигурацию** системы соответствующих функциональных компонент, учитывающих особенности каждого устройства. Такие компоненты называются **системными драйверами**, их полный перечень насчитывает около 100 программ, предназначенных для обслуживания различных марок клавиатур, адаптеров дисплея и принтеров:

- **драйверы дисплея** для изображения букв национального алфавита на экране дисплея;
- **драйверы дисплейного шрифта** задают начертания букв национального алфавита на экране дисплея;
- **драйверы клавиатуры** для переключения клавиатуры из режима ввода букв латинского алфавита в режим ввода букв национального алфавита;
- **драйверы клавиатурной раскладки** устанавливают соответствие между клавишами и буквами, вводимыми при их нажатии, а также задают способ кодирования букв национального алфавита;
- **драйверы принтера** обеспечивают вывод букв национального алфавита на принтер;
- **драйверы принтерного шрифта** задают начертание букв национального алфавита, выводимых на принтер.

Beta Plus представляет собой комплекс резидентных программ (т.е. программ, остающихся в оперативной памяти в ходе всего сеанса работы на ПК), в состав которого входит **основная управляющая программа**, набор **системных драйверов**, вспомогательная программа, предназначенная для включения нужных драйверов в рабочую конфигурацию системы (эта программа называется **загрузчиком драйверов**), и спра-

вочная программа, выдающая сведения о сформированной конфигурации. Разработаны драйверы, обеспечивающие поддержку русских букв на следующих видах клавиатур, адаптеров дисплеев и принтеров.

Клавиатуры: стандартная клавиатура зарубежных компьютеров (раскладка ЙЦУКЕНГ/QWERTY, для русских букв также может использоваться раскладка, принятая на отечественных пишущих машинках, кодировка основная и альтернативная), расширенная клавиатура зарубежных компьютеров (раскладка ЙЦУКЕНГ/QWERTY, для русских букв также может использоваться раскладка, принятая на отечественных пишущих машинках, кодировка основная и альтернативная), клавиатура ЕС1840/41/45 (раскладка ЙЦУКЕНГ/QWERTY, кодировка альтернативная).

Адаптеры дисплея: CGA, EGA, VGA (адаптер дисплея ПК PS/2 [50, 60, 70, 80]), MCGA (адаптер дисплея ПК PS/2 [25, 30]), Hercules Plus, графический адаптер ПК ЕС1840/41/45, монохромный адаптер ПК ЕС1840/41/45, адаптер дисплея ПК Olivetti M-24/28, адаптер дисплея ПК Toshiba LapTop, адаптер дисплея ПК IBM Convertible.

Принтеры: Epson FX-80, Epson FX-85/FX-100/FX-1000, Epson FX-800/850/1050, Epson LX-80, Proprinter (Europe), Proprinter II (Europe), Proprinter XL (Europe), Proprinter (USA), Proprinter II (USA), Proprinter XL (USA), Juki 5500/5510/5520, RAVI 8010M, Robotron-6329, Datamate DT-130, Taxan KP815/KP915, Star NX-1000, Citizen-1200, Epson LQ, Epson LQ-500, Proprinter X24, Proprinter XL24, Fujitsu DL 3300, NEC Pinwriter P6/P7, Amstrad LQ 3500, Quiet Writer II, Quiet Writer III, HP Laser Jeb II, HP Laser Jeb Plus) и др. На всех перечисленных принтерах русский алфавит может использоваться как в основной, так и в альтернативной кодировках, все принтеры оснащены шрифтами для печати.

ЛИТЕРАТУРА

1. Norton P. Programmer's Guide to the IBM PC. - Microsoft Press, 1985.
2. Shipman C. Color and Graphics. - HP Books Ins., 1984.
3. PC/AT Technical Reference. - IBM, 1984.
4. FX-80. Operation Manual. - Epson Corporation, 1982.

Н.Л. Глonti, С.В. Клименко, В.К. Малышев,
А.В. Самарин, Б.Б. Филимонов

МЕТАПРОЕКТ КИРИЛЛОВСКОГО АЛФАВИТА ДЛЯ ПЕЧАТАЮЩИХ УСТРОЙСТВ С ВЫСОКИМ РАЗРЕШЕНИЕМ

*“Одни только буквы и являются
необходимым и достаточным элементом
книги, все прочее же существует
благодаря им” [11].*

Джамбатиста Бодони

*“Без красивого шрифта нет
красивой книги, без красивой книги нет
настоящей культуры” [10].*

ВВЕДЕНИЕ

В XV веке Иоганн Гутенберг изобрел способ печати с помощью подвижного металлического шрифта, который использовался без каких-либо существенных изменений вплоть до XX века. Гутенберг и его первые последователи были мастерами на все руки: они разрабатывали и отливали шрифты, выполняли функции наборщиков, редакторов и издателей одновременно. Впоследствии потребности расширившегося книжного рынка привели к необходимости специализации и разделения труда. Постепенно издательское дело приобрело ту технологическую форму, которая дошла до наших дней: автор, издатель, художник-шрифтовик, изготовитель шрифтов, наборщик – каждый выполняет свою функцию.

Во второй половине нашего столетия поистине революционные преобразования в технику печати внесли электроника и вычислительная техника. Современные печатающие устройства, обладающие достаточной разрешающей способностью, позволяют получать издания

с высоким полиграфическим качеством. Теоретически каждому, кто пользуется таким оборудованием, доступны, подобно Гутенбергу и другим первопечатникам 500 лет назад, все стадии процесса изготовления книг — не только набор, редактирование и печать текста, но даже производство типографских шрифтов.

Созданная известным американским ученым Дональдом Кнудом система METAFONT [1] предназначена для проектирования шрифтов, используемых системами подготовки текстов. Одним из наиболее известных текстовых процессоров является T_EX [2], также разработанный профессором Д.Кнудом. T_EX и METAFONT выполняют две основные задачи при наборе текста: T_EX помещает его в нужные положения на странице, а METAFONT определяет начертания самих символов.

В основу METAFONT'a легла идея трех П: использование *перьев* и *параметров* в *программах*. На этой идее базируются три основные характерные особенности необычной системы Д.Кнута: 1) формы вычерчиваются воображаемыми перьями, имеющими некую *толщину*; 2) язык METAFONT'a поощряет использование параметров при проектировании рисунков символов, что позволяет описывать семейства форм; 3) описания буквоформ задаются в программах с учетом первых двух особенностей.

Программы на языке METAFONT'a весьма отличаются от обычных программ, так как они скорее “декларативны”, чем “императивны”. Иными словами, они описывают взаимоотношения между частями проекта, не предписывая, как эти взаимоотношения выполнять. Например, можно указать, что некоторая точка лежит на пересечении двух линий, но не объяснять, как вычислить ее координаты. METAFONT умеет решать эту и другие более сложные проблемы “за кулисами”.

METAFONT способен реализовать почти любое намерение художника, проектирующего шрифты. Система делает компьютер послушным инструментом дизайнера, используя его как кисть или резец. Более того, METAFONT может стать не просто пассивным

посредником, но и соавтором проекта: определяя важнейшие детали, можно предоставить ему инициативу, оставляя за собой окончательное решение.

Применение замечательной системы Д.Кнута для проектирования типографских шрифтов открывает новые перспективы в развитии современной типографии, укрепляя “мост между двумя культурами” и вовлекая в сотрудничество художников и ученых. Лучшие изделия METAFONT’a, вероятно, будут продуктом совместной работы специалистов обеих областей знаний.

Следует особо отметить, что METAFONT позволяет создавать “меташрифты” – схематическое описание всех символов алфавита, основанное на использовании заданных параметров. Очертания символов соответственно меняются, когда изменяются их основные параметры. Метапроектирование гораздо сложнее, чем обыкновенное проектирование: ведь легче начертить что-либо, чем объяснить, как чертить. Метапроект шрифта состоит из метаописаний отдельных его символов. Метаописание – это не рисунок символа, а набор описаний тех действий, которые нужно произвести, чтобы получить его изображение. Одной из проблем при разработке описания форм символов является то, что нелегко одновременно представить себе и специфицировать все потенциальные возможности метатехнологии. Однако когда получено удачное описание формы в достаточно общем виде, его можно использовать для изображения подобных форм в различных обстоятельствах.

Параметризуемость делает возможным получение на основе общего описания форм символов полной гарнитуры шрифта, т.е. всех типографских размеров для разновидностей шрифта, сохраняющих стиливые признаки основного рисунка алфавита.

Имея описание шрифта в виде программ на языке METAFONT, можно получить шрифты для печатающих устройств с любым разрешением. При этом качество шрифта тем выше, чем большей разрешающей способностью обладает устройство. Использование лазерного принтера с разрешением 300 точек/дюйм позволяет

воспроизводить текст, не уступающий по чистоте и четкости шрифта типографскому. Такие возможности накладывают определенную ответственность на разработчика шрифта, поскольку с помощью METAFONT'a легко производить шрифты низкого качества, а тщательное описание всех деталей красивой буквы — дело чрезвычайно тонкое. При этом разработчик должен обладать не только опытом работы с METAFONT'ом, но и определенным художественным чутьем и некоторыми сведениями из области полиграфии.

Появление в нашей многоязычной (многоалфавитной) стране современных печатающих устройств с высоким разрешением, а также мощных систем, предназначенных для набора и форматирования текстов и позволяющих использовать национальные алфавиты, делает необходимой своевременную подготовку компьютерных шрифтов высокого качества для всех письменностей СССР. В этом плане задачей первостепенной важности является разработка русского алфавита (кириллицы) — основы книгопечатания не только для русского, но и для некоторых других языков.

Рассмотрим основные принципы проектирования и разработки конкретного меташрифта на примере кириллицы. Поскольку инструментальным средством проектирования мы выбрали METAFONT, то приведем основные сведения о METAFONT'е, следуя работе [1].

С ЧЕГО НАЧИНАЕТСЯ ШРИФТ

Приступая к проектированию шрифта, прежде всего следует уяснить себе, каким должен быть основной рисунок, на основе которого будут получены различные варианты шрифта (с засечками, без засечек, различной плотности и насыщенности). После этого можно перейти к определению пропорций для каждой литеры. Здесь облегчает задачу использование метода полиграмм [7], основанного на объединении букв по группам. По основной литере такой группы опре-

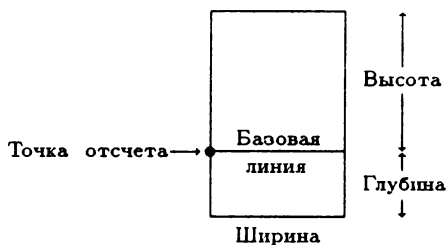
деляются пропорции и строятся некоторые элементы остальных. Например по прописной букве Н строятся буквы И, П, Ц, а также частично Б, В, Е, К, Р, Ю, Я. Желательно подготовить предварительный набросок для каждой литеры на листе миллиметровой бумаги.

Теперь можно приступить к описанию начертаний литер на языке METAFONT'a. Прежде всего, нужно сообщить METAFONT'у сведения о символе, необходимые для работы форматирующих программ: в первую очередь, код символа в таблице ASCII, а также размеры некоего прямоугольника — бокса, в который должен быть помещен символ.

БОКСЫ

Система набора текстов типа T_EX предполагает, что каждый символ заключен в прямоугольный бокс. T_EX никогда не заглядывает внутрь бокса, чтобы идентифицировать символ, — его задача состоит в том, чтобы расположить боксы в правильных местах на странице, основываясь на их размерах. А решить, каким должен быть размер бокса, и создать символ внутри него должен разработчик шрифта.

Боксы — двумерные объекты, однако им приписывают три размера, потому что вертикальная составляющая делится на две величины: высоту (выше базовой линии) и глубину (ниже базовой линии). Горизонтальный размер называется шириной.



Боксы имеют также четвертый размер, называемый коррекцией курсива, который дает TeX 'у дополнительную информацию: выступает буква справа или нет. Коррекция курсива добавляется к ширине бокса. Все эти размеры должны быть заданы величинами, не изменяющимися для различных устройств.

КООРДИНАТЫ

При построении отдельных элементов символа необходимо отметить важные (ключевые) для них точки на предварительном наброске, выполненном на миллиметровой бумаге. Им приписываются любые подходящие номера. Правильный выбор ключевых точек позволяет получить именно те очертания символа, которые предполагал набросок. Иногда подобрать такие точки бывает довольно сложно и их положение приходится много раз менять, добиваясь желаемого результата, так как METAFONT имеет свое собственное представление о кривых, которые эти точки соединяют, и не всегда разработчик может с ним согласиться. Для задания ключевых точек METAFONT использует декартовы координаты.

Единицей длины в METAFONT'е является пиксел. Например, точка $(x, y) = (100, 200)$ находится на 100 пикселей справа от начала координат и на 200 пикселей выше.

При составлении METAFONT-программ часто возникает необходимость использовать данные типа (x_k, y_k) , поэтому METAFONT допускает сокращение $z_k = (x_k, y_k)$.

Иногда пара координат рассматривается как вектор. Здесь векторы используются для указания направления движения при вычерчивании кривой. Например, вектор $(-1, 0)$ обозначает единицу движения влево (сокращенно 'left'). (Аналогично определяются единичные векторы 'right', 'up' и 'down').

Итак, сначала мы выбираем ключевые точки для изображаемого символа. Далее нужно указать META-

FONT'у их месторасположение внутри бокса, в котором будет размещен символ. Конечно, координаты точек можно задать непосредственно в пикселах, но это сделает программу менее наглядной. Кроме того, в процессе отладки, возможно, возникнет необходимость перемещения точки, что потребует новых вычислений. К тому же местонахождение точки может зависеть от различных обстоятельств, например от значений различных параметров, поэтому координаты точек обычно определяются системой уравнений с использованием переменных.

ПЕРЕМЕННЫЕ

К важнейшим понятиям METAFONT'а относятся переменные или типы данных, принимающие самые разнообразные значения. Одни из них, например 'x11' или 'y2', представляют собой координаты точек, другие, например 'right', — направления. Вообще же переменные здесь могут обозначать восемь разных типов данных:

- **boolean** (логическая переменная) принимает значения 'истинный' или 'ложный';
- **string** (строка) обозначает последовательность кодов ASCII;
- **path** (траектория) — линию или кривую;
- **pen** (перо) — форму наконечника рисующего инструмента;
- **picture** (изображение) — целую комбинацию пикселей;
- **transform** (преобразование) — операции масштабирования, поворота, сдвига или наклона;
- **pair** (пара) — два числа (например, точка или вектор);
- **numeric** (числовая величина).

Чтобы переменная представляла собой что-либо, кроме числа, нужно задать описание типа переменной. Для

этого указывают один из восьми типов, а затем перечисляют переменные, которые нужно описать. Например, описание

pair right, left

означает, что *right* и *left* — это переменные типа *pair*.

Нужно учитывать, что появление в программе описания обнуляет все предыдущие значения определяемых переменных.

Между переменными часто имеется неявная связь. Например, переменные типа *'slab#'* или *'bar#'* обозначают физические машинно-независимые расстояния, переменные *'slab'* и *'bar'* — соответствующие аналогичные машинно-зависимые расстояния в пикселах.

METAFONT позволяет легко составлять сложные имена, соответствующие взаимоотношениям между переменными. Например, переменную *'y[3].1'* можно обозначить просто как *'y31'*. Это очень удобно, поскольку координаты точек часто встречаются в программах.

Кроме локальных переменных, которые разработчик шрифта использует в программе для каждого символа, имеются также глобальные переменные, общие для всего шрифта. Они задают стандартные для него величины: высоту и глубину символов, толщину штрихов, наклон символов и т.п. Например, переменная *'stem#'* обозначает ширину основного штриха для строчных букв, а переменная *'cap_height#'* — высоту прописных букв. Эти переменные представлены в машинно-независимых расстояниях. Среди специальных переменных есть логические переменные, определяющие характер шрифта: с засечками или без, с различной или одинаковой шириной символов.

Все эти переменные помещены в специальный файл параметров. Подбирая значения этих параметров, можно получить семейство родственных шрифтов на основе одного базового описания всех символов.

Кроме того, METAFONT использует еще и так называемые “внутренние” величины, которые опре-

деляют поведение программы в конкретных случаях. Так, например, значение внутренней величины 'tracingequations' определяет, показывать или не показывать значения переменных по мере того, как они становятся известными. Все внутренние величины принимают только числовые значения.

Для выражения не только числовых, но и графических отношений между переменными METAFONT располагает богатым набором операций. Кроме элементарных арифметических действий, тригонометрических функций, логарифмов и экспонент в выражениях можно использовать множество менее привычных, но достаточно полезных операций. Например, операция $\frac{1}{3}[z_1, z_2]$ находит точку, лежащую на третьей части расстояния от z_1 до z_2 . Вообще,

$$t[z_1, z_2] = z_1 + t(z_2 - z_1).$$

Имеются также операции типа 'max' или 'min', вычисляющие минимум или максимум среди чисел, строк или пар и разнообразные операции округления. Тригонометрические функции в явном виде употребляются в METAFONT-программах довольно редко, так как в METAFONT'е есть более удобные для разработчика шрифтов функции 'dir' и 'angle'. Операция 'dir x ' задает вектор под углом x градусов по отношению к оси абсцисс, а операция 'angle(a,b)' определяет угол, соответствующий данному вектору.

Имеются операции над строками, парами чисел, траекториями, перьями и другими типами переменных. Например операция 'z shifted (2,3)' дает точку, лежащую на два пиксела вправо от точки z и на три пиксела выше ее, а операция 'point of pl' задает начальную точку траектории pl .

УРАВНЕНИЯ

Переменные в METAFONT-программах принимают значения, определяемые уравнениями, выражающими отношения между ними. Для координат ключевых точек это — их расположение относительно друг друга и границ бокса, в который помещен символ. Например, уравнение $z_4 = .5[z_5, z_6]$ определяет положение точки z_4 относительно точек z_5 и z_6 .

Чтобы определить координаты пяти ключевых точек, необходимо написать 10 уравнений, потому что каждое уравнение позволяет определить одно значение, а пять точек имеют 10 координат. Тут следует учитывать, что уравнение между парами координат (такое, как приведенное выше) считается за два уравнения между отдельными величинами.

Уравнения могут быть заданы в любом порядке. METAFONT всегда вычисляет значения стольких переменных, сколько возможно, основываясь на предыдущих уравнениях. Например, после двух уравнений

$$\begin{aligned} a + b + 2c &= 3, \\ a - b - 2c &= 1 \end{aligned}$$

METAFONT'у уже известно значение $a = 2$, а про b и c он знает, что $b + 2c = 1$.

В любой точке программы переменная “известна” или “неизвестна” в зависимости от того, можно или нельзя единственным образом вывести ее значение из составленных до сих пор уравнений.

METAFONT имеет дело с двумя типами неизвестных числовых переменных: зависимыми и независимыми. Каждая переменная в начале своего существования — независимая, но каждое новое уравнение, появляясь в программе, заставляет одну из независимых переменных стать зависимой или известной. Когда независимая переменная v меняется на зависимую или известную, эквиваленты всех зависимых переменных обновляются, так что они больше

не зависят от v , в этом процессе многие или все они меняются с независимых на известные.

Когда METAFONT обрабатывает числовое уравнение, он заменяет все известные переменные числовыми значениями, а все зависимые переменные — их эквивалентами. Получающееся в результате уравнение может быть преобразовано в

$$c_1 v_1 + \dots + c_m v_m = \alpha,$$

где все c — ненулевые константы, а все v — независимые переменные; α — числовая константа, которая может быть нулем. Если $m = 0$, то уравнение считается либо избыточным, если $\alpha = 0$, либо несовместным (в противном случае). Но если $m > 0$, METAFONT выбирает независимую переменную v_k , для которой c_k — это максимум, и переписывает уравнение в

$$v_k = (\alpha - c_1 v_1 - \dots - c_{k-1} v_{k-1} - c_{k+1} v_{k+1} - \dots - c_m v_m) / c_k.$$

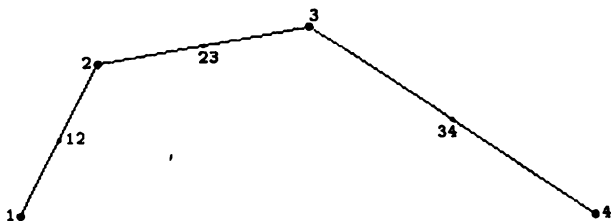
Переменная v_k становится зависимой, если $m > 1$, или известной, если $m = 1$.

Несовместные уравнения — это уравнения, не имеющие решений. Уравнения, которые не сообщают ничего нового, — избыточны. Например, $0 = 0$ избыточно так же, как и $a = b + c$, если ранее было задано $c = a - b$.

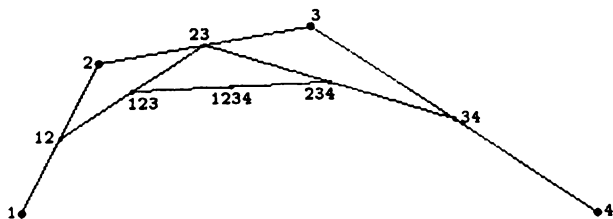
Значения переменных обычно вычисляются из уравнений. Но есть также другой способ, когда вместо знака '=' используется ':='. Операция присвоения ':=' означает, что отбрасывается предыдущее значение переменной и ей присваивается новое значение. Присвоения разрешаются только тогда, когда выражение слева от знака ':=' является переменной. Например, если в программе встречается запись ':=' , то это означает, что вычисленное ранее значение переменной x_2 теряется и заменяется значением координаты x_0 .

КРИВЫЕ

Определив месторасположение ключевых точек, можно приступить к вычерчиванию траекторий, которые должны их соединять. Для построения кривых METAFONT использует следующий *четырёхточечный* метод. Для заданных четырех точек (z_1, z_2, z_3, z_4) находятся три промежуточные точки $(z_{12} = \frac{1}{2}[z_1, z_2], z_{23} = \frac{1}{2}[z_2, z_3], z_{34} = \frac{1}{2}[z_3, z_4])$:

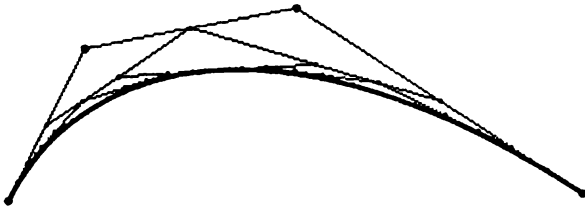


Затем для этих точек нужно построить две промежуточные точки второго порядка $z_{123} = \frac{1}{2}[z_{12}, z_{23}]$ и $z_{234} = \frac{1}{2}[z_{23}, z_{34}]$ и, наконец, промежуточную точку третьего порядка $z_{1234} = \frac{1}{2}[z_{123}, z_{234}]$:



Эта точка — одна из точек (z_1, z_2, z_3, z_4) , определяющих кривую. Чтобы получить остальные точки этой кривой, нужно повторять эти построения на точках

$(z_1, z_{12}, z_{123}, z_{1234})$ и $(z_{1234}, z_{234}, z_{34}, z_4)$ до бесконечности:



Полученная кривая обладает следующими важными свойствами:

- начинается в z_1 , продолжаясь в направлении от z_1 в z_2 ;
- заканчивается в z_4 , продолжаясь в направлении от z_3 в z_4 ;
- полностью остается внутри выпуклого каркаса z_1, z_2, z_3 , и z_4 .

Кривая, определяемая этими правилами, может быть описана с помощью формулы

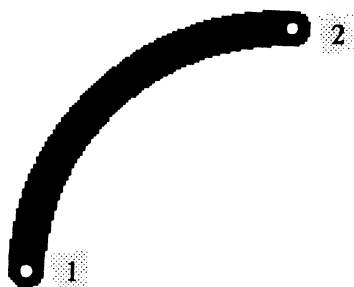
$$z(t) = (1-t)^3 z_1 + 3(1-t)^2 t z_2 + 3(1-t) t^2 z_3 + t^3 z_4,$$

где параметр t изменяется от 0 до 1. Этот полином третьей степени относительно t называется полиномом Бернштейна.

Все кривые, которые чертит METAFONT, основаны на четырех точках, но разработчику шрифта не обязательно указывать все эти точки, потому что METAFONT обычно сам может вычислить подходящие значения z_2 и z_3 . Обычно в программе на языке METAFONT'a указываются только конечные точки z_1 и z_4 , через которые предполагается провести кривую.

Управлять построением кривой можно, сообщая METAFONT'у, в каком направлении нужно двигаться в нескольких или во всех ключевых точках. Такие направления задаются с помощью векторов. Например,

кривая



получена изображением траектории

$z_1\{up\} \dots \{right\} z_2$.

Символы, соединяющие ключевые точки при задании траектории, указывают METAFONT'у характер кривой между этими точками. Таких обозначений всего пять:

- .. — свободная кривая;
- ... — ограниченная кривая;
- — прямая линия;
- — “натянутая” линия;
- & — место соединения.

Получив заданную траекторию ‘ $z_0 \dots z_1 \dots$ (и т.д.) $\dots z_{n-1} \dots z_n$ ’, METAFONT вычисляет кривую длиной n , которая соответствует его представлениям о “наиболее красивой кривой”, проходящей через данные точки от z_0 до z_n .

Обозначение ‘...’ в сущности то же самое, что и ‘..’, за исключением того, что оно заставляет траекторию придерживаться ограничивающего треугольника, определяемого конечными точками и направлениями. Предположим, что кривая идет от z_0 к z_1 и существует точка z , такая, что кривая начинает движение в направлении от z_0 к z и заканчивается, двигаясь

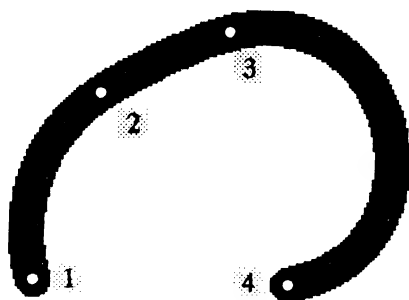
от z к z_1 . Тогда кривая, указанная с помощью '...', будет полностью оставаться внутри треугольника с вершинами в z_0 , z_1 и z . Но если такого треугольника нет, то '...' и '..' дадут одинаковые кривые.

Сегмент прямой линии ' $z_{k-1} -- z_k$ ' обычно заставляет траекторию круто изменить направление в z_{k-1} и z_k . И наоборот, сегмент, указанный ' $z_k --- z_{k-1}$ ', будет прямой линией, которая плавно переходит в соседние кривые, т.е. траектория будет входить в z_{k-1} и выходить из z_k в направлении $z_k - z_{k-1}$.

И, наконец, операция '&' соединяет две независимые траектории в общей точке, так же как '&' соединяет две строки.

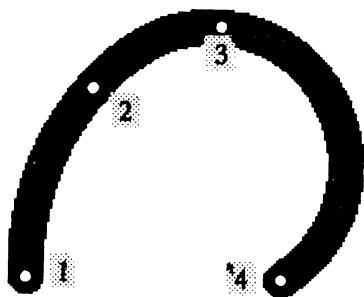
Есть еще много различных способов движения из одной точки в другую, основанных на указании так называемых "натяжений" или "закруглений", позволяющих изменять форму кривой в конечных точках. Например, кривая, полученная изображением траектории

$$z_1 \dots z_2 \dots \text{tension } 1.2 \dots z_3 \dots z_4,$$



"стянута" между точками z_2 и z_3 , если сравнить ее с кривой

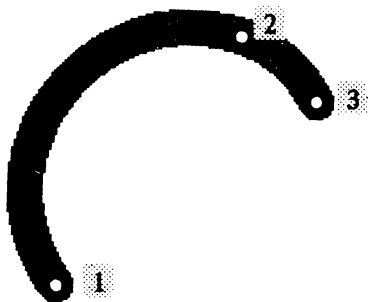
$$z_1 \dots z_2 \dots \text{tension } 1 \dots z_3 \dots z_4,$$



где натяжение между всеми точками по умолчанию равно 1.

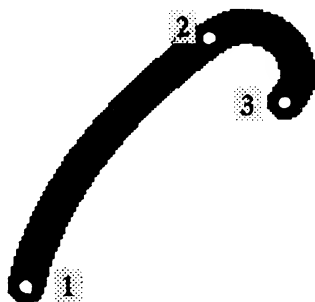
Указав различную величину переменной закругления (curl), по умолчанию также равной 1, можно получить разные кривые. Например, кривая

$$z_1\{\text{curl}0\} \dots z_2 \dots \{\text{curl}0\} z_3,$$



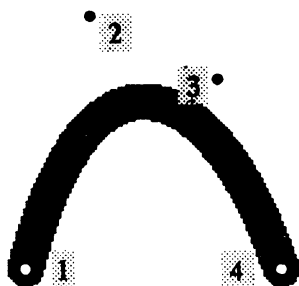
отличается от кривой

$$z_1\{\text{curl}2\} \dots z_2 \dots \{\text{curl}2\} z_3:$$



Если все вышеперечисленные способы недостаточны для получения требуемой траектории, в качестве последнего средства всегда можно указать все четыре точки в четырехточечном методе. Например, кривая

$z_1 \dots \text{controls } z_2 \text{ and } z_3 \dots z_4$



начинается в точке z_1 , заканчиваясь в z_4 , а точки z_2 и z_3 являются контрольными.

METAFONT умеет оперировать с переменными типа **path**, так что есть возможность получать новые траектории из уже построенных. Можно не только объединять кривые, но и сдвигать, масштабировать, поворачивать и исправлять разнообразными способами. Например, запись

$p \text{ rotated around } (z_0, 35)$

задает траекторию p , повернутую на 35° вокруг точки z_0 .

Возможно также выделить ту часть траектории, которая нам нужна, или найти точку, лежащую на данной кривой. Для этого необходимо ввести понятие длины траектории и движения по ней. Длина траектории — это количество шагов '.', которые она содержит. Например, длина траектории $p = z_0 \dots z_1 \dots z_2$ равна 2.

Выражение 'point t of p ' задает положение точки, движущейся по траектории p , начиная от начальной точки z_0 в момент времени $t = 0$ и достигая точки

z_2 при $t = 2$. Значение $t = 0.5$ дает среднюю точку на траектории между z_0 и z_1 .

Запись 'subpath (1,2) of p ' определяет субтраекторию $z_1 \dots z_2$.

Весьма важной операцией, которую METAFONT может производить с траекториями, является пересечение. Если даны две траектории p и q , то

p intersectiontimes q

дает пару моментов времени (t, u) , таких, что

point t of p = point u of q .

Аналогичная операция

p intersectionpoint q

показывает уже не моменты времени, а общую точку пересечения.

ПЕРЬЯ

Для вычерчивания траекторий METAFONT использует воображаемые перья с наконечниками разнообразных форм и размеров. Выбор пера осуществляет команда `pickup`. Например, команда

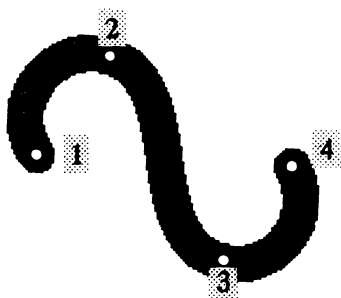
`pickup pencircle scaled 0.4pt`

дает указание использовать круглое перо `pencircle` шириной в один пиксел, умножив его диаметр на $0.4pt$, где pt — количество пикселей в точке (дискрете) принтера, когда в расчет принимается текущее разрешение.

Используются также перья с овальными или прямоугольными наконечниками, бесконечно тонкое перо. Вообще, можно определить перо любой выпуклой

многоугольной формы, задав эту форму циклической траекторией.

Когда METAFONT получает задание изобразить что-либо, например с помощью команды **draw**, он использует текущее перо, т.е. перо, которое было выбрано до этого. Например, команда '**draw** $z_1 \dots z_2 \dots z_3 \dots z_4$ ' соединяет точки z_1 , z_2 , z_3 и z_4 линией, проведенной текущим пером:



От формы пера зависит толщина линии и ее очертания в конечных точках траектории.

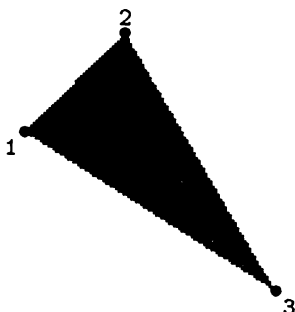
Способность METAFONT'a чертить, используя команду **draw**, позволяет создавать самые разнообразные формы символов, но здесь есть существенное ограничение: смоделированный кончик пера должен оставаться одинаковым на всем протяжении проводимой линии.

Команда METAFONT'a **fill** позволяет получить более точное изображение. Она закрашивает область, ограниченную замкнутой траекторией. Например, команда

fill $z_1 \text{ -- } z_2 \text{ -- } z_3 \text{ -- cycle}$

изображает закрашенный треугольник с вершинами в

точках z_1 , z_2 и z_3 :



Возможность заполнения контуров позволяет вообразить себе, что мы имеем перья с широкими концами, направление и нажим которых меняются, когда они скользят по бумаге, если рассматривать отдельные траектории, проведенные левым и правым концами пера. Положения такого пера в ключевых позициях определяют точки z_{1l} , z_1 , z_{1r} , обозначающие соответственно левый край, центр и правый край пера. Взаимоотношения между такими тройками точек устанавливаются командой *'penpos'*, определяющей ширину пера и угол его наклона. Например, команда

penpos₁(0.8pt, 45)

указывает, что в точке z_1 перо имеет ширину 0.8pt и наклонено к базовой линии на 45° .

Однако следует отметить, что эти смоделированные перья имеют серьезное ограничение по сравнению с настоящим каллиграфическим пером: левый и правый концы пера, сделанного с помощью *'penpos'*, никогда не должны пересекаться. Поэтому нужно поворачивать перо при движении вдоль кривой.

Имеется также комбинация команд **fill** и **draw**. Это команда **filldraw**, которая закрашивает область, ограниченную траекторией, изображенной текущим пером.

ИЗОБРАЖЕНИЯ

METAFONT имеет свою внутреннюю графическую бумагу, которая образует так называемый растр или решетку, состоящую из квадратных пикселей (квадратный пиксел имеет ширину и высоту в один пиксел.) Создавая пиксельную картинку символа, METAFONT будет указывать, что некоторые пиксели “черные”, а другие, “белые”, т.е. по существу формы преобразуются в двоичные комбинации. Частота решетки называется ее разрешением, единицей обычно является пиксел/дюйм (в США) или пиксел/миллиметр (в “метрических” странах).

Когда METAFONT получает задание начертить кривую или заполнить область, он закрашивает ограниченные заданной траекторией пиксели. На самом деле это означает, что изменяются (увеличиваются на 1) некие целые значения, соответствующие закрашенным пикселям, а значения незакрашенных пикселей остаются неизменными. При получении изображения черными пикселями будут те, значение которых больше нуля. Некоторые пиксели могут иметь значение больше 1, потому что они заполнены несколько раз.

Такая организация внутренней графической бумаги METAFONT'a позволяет складывать и вычитать изображения, суммируя или вычитая значения соответствующих пикселей. Изображения можно также сдвигать, отражать и поворачивать на число градусов, кратное 90.

ОТ ОДНОГО СИМВОЛА К ПОЛНОМУ ШРИФТУ

Посмотрим теперь, как выглядит описание на языке METAFONT конкретного символа шрифта. Ниже приведена METAFONT-программа, изображающая прописную русскую букву Ф (см. рисунок).

```
1 cmchar "The cyrillic letter F";
2 beginchar(oct"146",16u#,cap_height#,0);
```

```

3 italiccorr .75cap_height#*slant-.5u#;
4 pickup tiny.nib;
5 penpos1(cap_stem,0); penpos2(cap_stem,0);
6 x1=x2=.5w; top y1=h; bot y2=0;
7 filldraw stroke z1e--z2e; % основной штрих
8 penpos3(cap_band,90); penpos4(curve,180);
9 penpos5(cap_band,-90); penpos6(curve,0);
10 x3=x5=x1; x4r=w-x6r=.5u;
11 y3r=h-y5r=.8h; y4=y6=.5top[y3,y5];
12 fill stroke pulled_arc.e(3,4)
13 & pulled_arc.e(4,5) & pulled_arc.e(5,6)
14 & pulled_arc.e(6,3); % округлая часть
15 if serifs:
16 dish_serif(1,2,a,1/3,cap_jut,b,1/3,cap_jut);
17                                     % верхняя засечка
18 dish_serif(2,1,c,1/3,cap_jut,d,1/3,cap_jut);fi
19                                     % нижняя засечка
20 penlabels(1,2,3,4,5,6); endchar;

```

В первой строке программы сообщается название символа, которое затем используется в пробных распечатках. Далее в каждой программе, определяющей символ, должна быть выполнена команда

beginchar(*(code)*, *(width)*, *(height)*, *(depth)*),

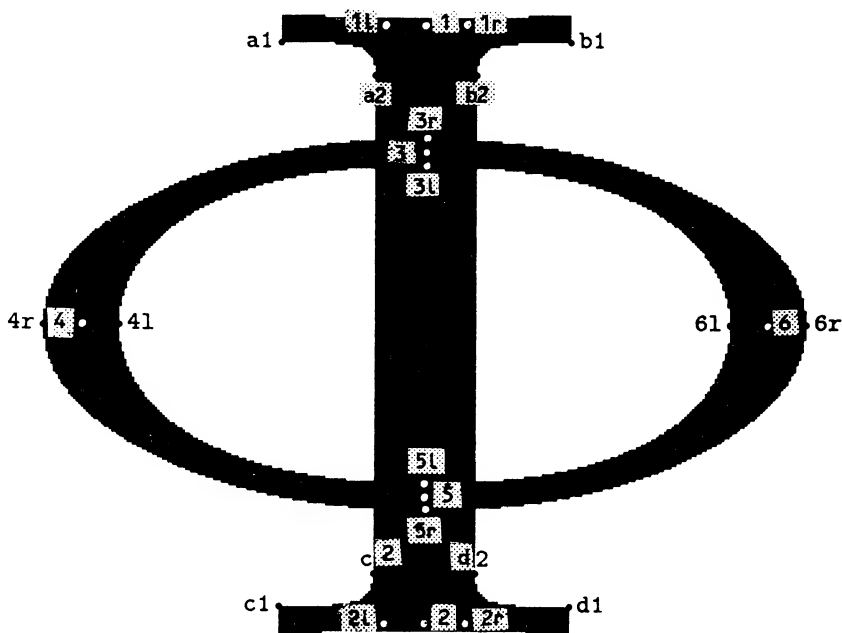
где *(code)* – ASCII-код символа, а три другие величины – *(width)*, *(height)* и *(depth)* – показывают ширину, высоту и глубину прямоугольного бокса, в который заключена литера. Операция **beginchar** присваивает эти значения соответственно специальным переменным *w*, *h* и *d*.

Строка 3 задает коррекцию курсива, т.е. величину, которая добавляется к ширине бокса, если символ изображается наклонным шрифтом.

В строке 4 осуществляется выбор уже существующего стандартного пера, номер которого дает переменная *tiny.nib*.

В строке 5 определяются ширина и наклон пера – в ключевых точках 1 и 2, а затем в строке

6 даются уравнения, показывающие положения этих точек в боксе. Информация, которая содержится в строках 5-6, достаточна для однозначного определения координат ключевых точек z_1 и z_2 , а также левых и правых краев пера, представленных вспомогательными точками z_{1l} , z_{1r} , z_{2l} и z_{2r} .



В строке 7 команда `filldraw` закрашивает текущим пером прямоугольную область, ограниченную отрезками прямых линий, последовательно соединяющих точки z_{1l} , z_{1r} , z_{2l} и z_{2r} , изображая основной штрих буквы Φ .

Далее, решая уравнения, приведенные в строках 8-11, METAFONT вычисляет координаты остальных ключевых и вспомогательных точек литеры, на которых затем строятся дуги и закрашивается округлый элемент буквы Φ .

И, наконец, в строке 15 проверяется значение логической переменной *serifs*. Если оно истинно, генерируется шрифт с засечками и подпрограмма 'dish_serifs' изображает эти характерные детали в точках 1 и 2.

Команда **penlabels** позволяет увидеть метки перечисленных в ней ключевых и вспомогательных точек на пробных распечатках увеличенных копий буквы.

Программу для каждой литеры должна завершать команда **endchar**, которая записывает изображение символа в выходной файл или, возможно, отображает его на экране.

Эта программа несколько упрощена по сравнению с той, которая вошла в описание шрифта, разработанного нами с помощью METAFONT'a, но все же дает представление об основных принципах проектирования шрифтов. Пропустив программу через METAFONT, всякий раз изменяя значения параметров, можно получить множество различных вариантов изображения символа. Вот некоторые из них:

Ф Ф Ф Ф Ф Ф

Ф Ф Ф Ф Ф Ф

Ф Ф Ф Ф Ф Ф

Таким же образом были разработаны программы для всех прописных и строчных букв кириллицы. Тексты программ содержатся в файлах, именами которых удобно идентифицировать соответствующие шрифты (прописные CYRU.MF, строчные CYRL.MF). Была подготовлена также METAFONT-программа для курсивного (*italic*) начертания строчных букв (файл CITALL.MF).

При проектировании представляемого шрифта учтена возможность получения алфавитов разных стилей

посредством изменения специальных параметров. А именно, используя перечисленные программы, можно получить следующие шрифты:

CMCYR (Computer Modern Cyrillic) – светлое нормальное начертание;

CMCSL (Computer Modern Cyrillic Slanted) – светлое нормальное начертание с наклоном;

CMCBX (Computer Modern Cyrillic Bold Extended) – жирное нормальное начертание;

CMCBXSL (Computer Modern Cyrillic Bold Extended Slanted) – жирное нормальное начертание с наклоном;

CMCSS (Computer Modern Cyrillic Sans Serif) – начертание без засечек (серифов);

CMCTT (Computer Modern Cyrillic Typewriter Text) – шрифт, используемый для отображения текста на экране терминала, все символы одинаковой ширины;

CMCTI (Computer Modern Cyrillic Text Italic) – светлое курсивное начертание.

У каждого из этих файлов столько различных модификаций, сколько типографских размеров (кеглей) используется для данного шрифта. Наиболее часто применяются так называемые “десятиточечные” шрифты, символы которых имеют условный размер в десять типографских пунктов. Эти шрифты можно получить, с помощью файлов параметров CMCYR10.MF, CMCBX10.MF, CMCSS10.MF и т.д.

Внешний вид разработанных шрифтов представлен в приложении.

Процессом генерации шрифта для печатающего устройства с конкретным разрешением управляет драйверный файл, который загружается файлом параметров, определяющим характер шрифта. В свою очередь, драйверный файл вызывает один или несколько

программных файлов. При генерации представляемого шрифта вызываются файлы с описанием русских прописных и строчных букв, а также файлы, содержащие программы для цифр и других сопутствующих символов шрифта. Драйверных файлов может быть несколько, и каждый из них использует различные комбинации программных файлов. Например, для разработанного шрифта подготовлены два драйверных файла: CYRILLIC.MF и TEXTIT.MF. Первый из них генерирует семейство обыкновенного, а второй — курсивного начертаний.

При генерации шрифта основной вывод METAFONT'a поступает в так называемый GF-файл (Generic Font). Этот файл содержит двоичное представление оцифрованного шрифта и некоторую дополнительную информацию, необходимую для программ, которые приводят в действие печатающее устройство. GF-файл легко преобразовать в любой другой цифровой формат шрифта. С его помощью можно получить распечатки увеличенных копий символов либо при окончательной генерации шрифта — пиксельное представление всех литер.

Второй тип выходных файлов METAFONT'a — это файлы TFM (T_EX Font Metric), которые содержат информацию о шрифте, используемую форматизирующей программой T_EX. Файл TFM сообщает T_EX'у размеры боксов, в которых размещены литеры, а также коррекцию курсива для каждого из них. Следующий вид информации, необходимой T_EX'у для качественного набора текста, связан с парами соседних символов, которые набираются из одного шрифта. Качество набора текста во многом зависит от правильной установки межсимвольных расстояний. Межсимвольный пробел образуется из двух полупробелов поставленных рядом литер. Поскольку символы имеют различную форму, то одинаковые для всех литер межсимвольные пробелы будут оптически восприниматься по-разному: одни будут казаться увеличенными, а другие, напротив, уменьшенными. Поэтому для некоторых сочетаний символов следует уменьшить или, наоборот, увеличить

межсимвольные расстояния. Эта информация задается в очень простых по форме `ligtable`-программах, которые включаются в драйверный файл. Например, программа

```
ligtable oct"101":                                % a
oct"121" kern -u#,                                % ia
oct"125" kern -u#,                                % u
oct"104" kern 2u#,                                % d
oct"136" kern -u#;                                % ch
```

сигнализирует системе набора текстов о необходимости уменьшить расстояние между буквами на единицу ширины `u#`, когда ей встречаются последовательности символов 'ая', 'ау' или 'ач', а для сочетания 'ад', наоборот, нужно увеличить межбуквенный пробел на две единицы.

Все метрические данные о шрифте, которые получает файл TFM, должны быть выражены в машинно-независимых единицах. Когда конкретный шрифт производится с различными режимами или увеличениями, то все его файлы TFM должны быть одинаковыми.

ЗАКЛЮЧЕНИЕ

Революция в книгопечатном деле, которая охватила весь современный мир, постепенно проникает и в нашу страну вместе с современной вычислительной техникой и высококачественными растровыми печатающими устройствами (электростатическими, лазерными и термопринтерами). Эти перемены неизбежно повлекут за собой появление настольных издательств, основное оборудование которых состоит из компьютера, оснащенного принтером. Качество готовой продукции современного настольного издательства во многом зависит от качества и разнообразия используемых шрифтов для подготовки оригинал-макета издания.

Революция в полиграфии в скором будущем проникнет во все республики СССР — отсюда очевидна

потребность в создании компьютерных шрифтов для национальных алфавитов нашей страны. Представленная разработка* одного комплекта кириллицы — лишь один из первых шагов на пути к решению этой насущной задачи.

В заключение авторы выражают благодарность Д.Вулису (CUNY, Department of Mathematics) за предоставленные материалы. Один из авторов (Н.Глонти) благодарит также руководство ИФВЭ и директора НТТУМ Министерства народного образования СССР К.Ш.Цискаридзе за поддержку и предоставленную возможность работы в ИФВЭ.

* Данная статья напечатана на лазерном принтере "HP LaserJet series II" с использованием девяти различных модификаций разработанных одним из авторов (Н.Глонти) шрифтов.

Приложение

1. CMCYR10 – светлое нормальное начертание

	'0	'1	'2	'3	'4	'5	'6	'7	
'04x		!		#	\$	%	&	'	"2x
'05x	(	)	*	+	,		.	/	
'06x	0	1	2	3	4	5	6	7	"3x
'07x	8	9	:	;		=		?	
'10x	ю	а	б	ц	д	е	ф	г	"4x
'11x	х	и	й	к	л	м	н	о	
'12x	п	я	р	с	т	у	ж	в	"5x
'13x	ь	ы	э	ш	э	щ	ч	ъ	
'14x	Ю	А	Б	Ц	Д	Е	Ф	Г	"6x
'15x	Х	И	Й	К	Л	М	Н	О	
'16x	П	Я	Р	С	Т	У	Ж	В	"7x
'17x	Ь	Ы	Э	Ш	Э	Щ	Ч	Ъ	
	"8	"9	"A	"B	"C	"D	"E	"F	

2. CMCSL10 – светлое нормальное начертание с наклоном

	'0	'1	'2	'3	'4	'5	'6	'7	
'04x		!		#	\$	%	&	'	"2x
'05x	(	)	*	+	,		.	/	
'06x	0	1	2	3	4	5	6	7	"3x
'07x	8	9	:	;		=		?	
'10x	ю	а	б	ц	д	е	ф	г	"4x
'11x	х	и	й	к	л	м	н	о	
'12x	п	я	р	с	т	у	ж	в	"5x
'13x	ь	ы	э	ш	э	щ	ч	ъ	
'14x	Ю	А	Б	Ц	Д	Е	Ф	Г	"6x
'15x	Х	И	Й	К	Л	М	Н	О	
'16x	П	Я	Р	С	Т	У	Ж	В	"7x
'17x	Ь	Ы	Э	Ш	Э	Щ	Ч	Ъ	
	"8	"9	"A	"B	"C	"D	"E	"F	

3. СМСВХ10 – жирное нормальное начертание

	'0	'1	'2	'3	'4	'5	'6	'7	
'04x		!		#	\$	%	&	'	"2x
'05x	(	)	*	+	,		.	/	
'06x	0	1	2	3	4	5	6	7	"3x
'07x	8	9	;	;		=		?	
'10x	ю	а	б	ц	д	е	ф	г	"4x
'11x	х	и	й	к	л	м	н	о	
'12x	п	я	р	с	т	у	ж	в	"5x
'13x	ь	ы	э	ш	э	щ	ч	ъ	
'14x	Ю	А	Б	Ц	Д	Е	Ф	Г	"6x
'15x	Х	И	Й	К	Л	М	Н	О	
'16x	П	Я	Р	С	Т	У	Ж	В	"7x
'17x	Ь	Ы	Э	Ш	Э	Щ	Ч	Ъ	
	"8	"9	"A	"B	"C	"D	"E	"F	

4. СМСВХSL10 – жирное нормальное начертание с наклоном

	'0	'1	'2	'3	'4	'5	'6	'7	
'04x		!		#	\$	%	&	'	"2x
'05x	(	)	*	+	,		.	/	
'06x	0	1	2	3	4	5	6	7	"3x
'07x	8	9	:	;		=		?	
'10x	ю	а	б	ц	д	е	ф	г	"4x
'11x	х	и	й	к	л	м	н	о	
'12x	п	я	р	с	т	у	ж	в	"5x
'13x	ь	ы	э	ш	э	щ	ч	ъ	
'14x	Ю	А	Б	Ц	Д	Е	Ф	Г	"6x
'15x	Х	И	Й	К	Л	М	Н	О	
'16x	П	Я	Р	С	Т	У	Ж	В	"7x
'17x	Ь	Ы	Э	Ш	Э	Щ	Ч	Ъ	
	"8	"9	"A	"B	"C	"D	"E	"F	

5. CMCSS10 – начертание без засечек (серифов)

	'0	'1	'2	'3	'4	'5	'6	'7	
'04x	.	!		#	\$	%	&	'	"2x
'05x	(	)	*	+	,		.	/	
'06x	0	1	2	3	4	5	6	7	"3x
'07x	8	9	:	;		=		?	
'10x	ю	а	б	ц	д	е	ф	г	"4x
'11x	х	и	й	к	л	м	н	о	
'12x	п	я	р	с	т	у	ж	в	"5x
'13x	ь	ы	э	ш	э	щ	ч	ъ	
'14x	Ю	А	Б	Ц	Д	Е	Ф	Г	"6x
'15x	Х	И	Й	К	Л	М	Н	О	
'16x	П	Я	Р	С	Т	У	Ж	В	"7x
'17x	Ь	Ы	Э	Ш	Э	Щ	Ч	Ъ	
	"8	"9	"A	"B	"C	"D	"E	"F	

6. CMCTT10 – шрифт, используемый для отображения текста на экране терминала

	'0	'1	'2	'3	'4	'5	'6	'7	
'04x		!		#	\$	%	&	'	"2x
'05x	(	)	*	+	,		.	/	
'06x	0	1	2	3	4	5	6	7	"3x
'07x	8	9	:	;		=		?	
'10x	ю	а	б	ц	д	е	ф	г	"4x
'11x	х	и	й	к	л	м	н	о	
'12x	п	я	р	с	т	у	ж	в	"5x
'13x	ь	ы	э	ш	э	щ	ч	ъ	
'14x	Ю	А	Б	Ц	Д	Е	Ф	Г	"6x
'15x	Х	И	Й	К	Л	М	Н	О	
'16x	П	Я	Р	С	Т	У	Ж	В	"7x
'17x	Ь	Ы	Э	Ш	Э	Щ	Ч	Ъ	
	"8	"9	"A	"B	"C	"D	"E	"F	

7. CMCT10 – светлое курсивное начертание

	'0	'1	'2	'3	'4	'5	'6	'7	
'04x		!		#	\$	%	&	'	"2x
'05x	(	)	*	+	,		.	/	
'06x	0	1	2	3	4	5	6	7	"3x
'07x	8	9	:	;		=		?	
'10x	ю	а	б	ц	д	е	ф	г	"4x
'11x	х	и	й	к	л	м	н	о	
'12x	п	я	р	с	т	у	ж	в	"5x
'13x	ь	ы	э	ш	э	щ	ч	з	
'14x	Ю	А	Б	Ц	Д	Е	Ф	Г	"6x
'15x	Х	И	Й	К	Л	М	Н	О	
'16x	П	Я	Р	С	Т	У	Ж	В	"7x
'17x	Ь	Ы	Э	Ш	Э	Щ	Ч	З	
	"8	"9	"A	"B	"C	"d	"E	"F	

ЛИТЕРАТУРА

1. Knuth D.E. The METAFONTbook. – Addison Wesley PC, 1984.
2. Knuth D.E. The T_EXbook. – Addison Wesley PC, 1984.
3. Knuth D.E. The concept of a meta-font // Visible language. – 1982.–N 16. – P. 3-27. .
4. Knuth D.E. Lessons learned from METAFONT // Visible language. – 1985. – N 19. – P. 35-53.
5. Beeton B. Matematical Symbols and Cyrillic Fonts Ready for Distribution (Revised) // TUGboat. – 1985. – V. 6, N 3. – P. 124-128.
6. Жильченков В.Д., Клименко С.В., Кочин В.Н., Павлов Ю.Г., Самарин А.В., Солдатов А.Ф., Соколов А.П. Автоматизированная система подготовки научно-технической документации PROTOTYP // Управляющие системы и машины. – 1984. – N 4.

7. *Спиров Н.А.* Проектирование и производство типографских шрифтов. - М.: Искусство, 1959.
8. *Гиленсон П.Г.* Справочник художественного и технического редакторов. - М.: Книга, 1988.
9. *Шрифты для художников-оформителей / Сост. Г.Ф.Кликушин.* - Минск: Полымя, 1980.
10. *Тоотс В.* 300 шрифтов. - Riga: Latvijas Valsts izdevnieciba, 1960.
11. *Книгопечатание как искусство / Сост. Р. Зиховски, Г. Тиман.* - М.: Книга, 1987.
12. *Немировский Е.Л.* Иоганн Гутенберг. - М.: Наука, 1989.

Тел.: 925-14-89 и 925-25-28, Протвино М.О.

КРАТКОЕ СООБЩЕНИЕ

УДК.681.325.5

**А.В.Кобылинский, В.В.Бестенков, Е.Я.Столяров,
В.А.Темченко**

8-РАЗРЯДНЫЙ МИКРОПРОЦЕССОР KP580BM1

Наиболее распространен в практике приборостроения на сегодняшний день МПК серии KP580. Описания систем на базе данной серии есть во многих учебных пособиях, монографиях и любительских журналах. Наименования приборов и систем на ее основе исчисляются тысячами, а потребность в ИС растет из года в год с геометрической прогрессией.

Однако хорошо разобравшись с особенностями применения в качестве ЦП МП KP580BM80A, многие потребители пришли к выводу, что возможности данного МП исчерпаны, и хотя существует огромный задел по рабочим местам отладки ПО, КИО, организации входного контроля, пакетам ППО, возникла необходимость использования в качестве ЦП других МП. К тому же применение трех источников питания для МП KP580BM80A существенно снижает технико-экономические и малогабаритные показатели аппаратуры и является просто анахронизмом. Возникает вопрос: имеет ли перспективы серия KP580?

В качестве ответа предлагается МП КР580ВМ1, который позволит МПК серии КР580 продолжить жизнь при внедрениях в новые разработки и серийную продукцию. При разработке МП КР580ВМ1 была поставлена и решена задача максимально избавиться от недостатков МП КР580ВМ80А при повышении технологичности, существенном улучшении эксплуатационных характеристик и расширении функциональных возможностей МП.

Это достигнуто за счет реализации в МП двух режимов работы:

первый - позволяет потребителям, которые в силу объективных причин не могут переработать изделие, существенно повысить эксплуатационные характеристики изделия в целом: увеличить быстродействие в 2,5 раза, уменьшить потребляемую мощность при сохранении всех пакетов ПП;

второй - возможность наряду с повышением аппаратного быстродействия увеличить производительность за счет расширения системы команд, адресации памяти до 128 Кбайт, использования периферийных БИС серии КР1810.

К указанным условиям следует добавить, что в новых применениях необходим генератор тактовых импульсов с одной серией тактовых сигналов (КР1810ВГ84).

Благодаря свойствам МП за счет реализации функций системного контроллера можно, применяя серийные ИС, реализовывать устройства при меньших аппаратных затратах.

Миросхема КР580ВМ1 - БИС однокристалльного высокопроизводительного 8-разрядного микропроцессора, предназначенная для использования в качестве центрального процессорного устройства при построении средств вычислительной техники - от простейших школьных микро-ЭВМ до мощных мультимикропроцессорных систем, систем управления технологическими процессами и производственным оборудованием, контрольно-измерительных систем. МП КР580ВМ1 разработан в развитие серии КР580 с сохранением наработанного матобеспечения.

Микросхема функционирует в двух режимах:

- обычном режиме эмуляции МП КР580ВМ80А (режим 0);
- расширенном режиме непосредственного формирования сигналов управления обменом (режим 1).

Структурная схема и условное графическое обозначение МП КР580ВМ1 приведены на рис.1 и 2, назначение выводов, сравнительные характеристики МП КР580ВМ80А и КР580ВМ1 и основные эксплуатационные параметры - в табл.1 - 3.

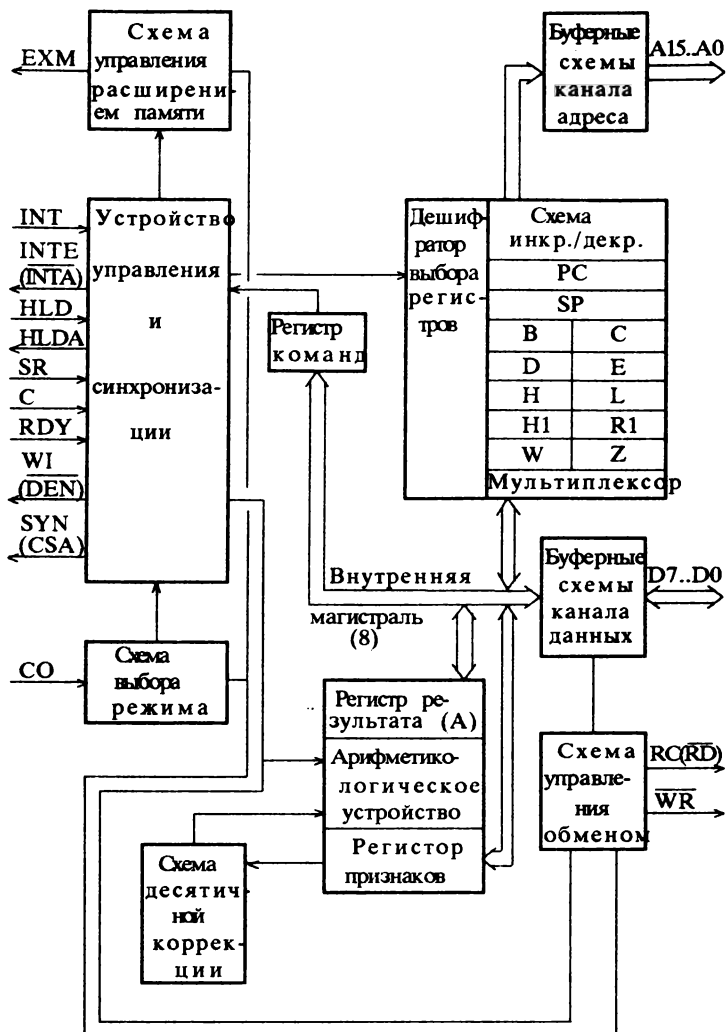


Рис. 1. Структурная схема МП KP580BM1 (в скобках указаны обозначения выводов в режиме 1)

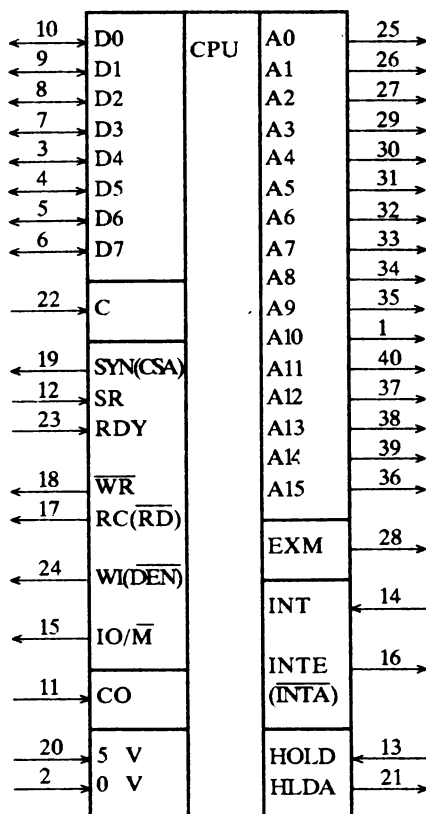


Рис.2 Условное графическое обозначение
МП KP580BM1

Особенности микросхемы:

- система команд совместима с системой команд МП KP580BM80A на уровне кодов команд;
- МП в одном из режимов аппаратно совместим с МП KP580BM80A;
- микросхема по входам и выходам совместима с микросхемами серий KP580, KP1810;
- в состоянии захвата шины адреса и данных и основные управляющие сигналы находятся в третьем состоянии, что позволяет применять МП в мультипроцессорных системах;

Таблица 1

Назначение выводов

Номер вывода	Наименование		Обозначение		Тип вывода		
	Режим 0	Режим 1	Режим 0	Режим 1	Режим 0	Режим 1	
1	Канал адреса		A10		Выход с тремя состояниями		
2	Общий		0V		—		
3...10	Канал данных		D4...D7, D3...D0		Вх/вл:х с тремя состояниями		
11	Управление		CO		Вход		
12	Сброс		SR				
13	Захват		HOLD				
14	Запрос прерывания		INT				
15	Признак обращения к УВВ или ЗУ		10/M				
16	Разрешение прерывания	Подтверждение прерывания	INTE	INTA	Выход	Вход с тремя состояниями	
17	Прием	Чтение	RC	RD			
18	Запись		WR				
19	Признак начального цикла	Строб состояния	SYN	CSA			
20	Напряжение питания		VCC		5 В		
21	Подтверждение захвата		HLDA		Выход		
22	Тактовый импульс		C		Вход		
23	Готовность		RDY				
24	Ожидание	Разрешение передачи данных	WI	DEN	Выход	Выход с тремя состояниями	
25—27	Канал адреса		A0 - A2		Не испол.		
28	Расширение памяти		Не испол.	EXM			
29...40	Канал адреса		A3...A5, A15, A12...A14, A11				

- для защиты от электрических перегрузок необходимо применять меры по защите микросхемы от статического электричества более 200 В.

Таблица 2

Сравнительные характеристики МП КР580ВМ80А и КР580ВМ1

Наименование параметра, единица измерения	КР580ВМ80А	КР580ВМ1
Напряжение питания, В	-5; + 5; + 12	+ 5
Тактовая частота, МГц	2,0	5,0
Объем адресуемой памяти, Кбайт	64	128
Число базовых команд	78	190
Формат команд, байт	1,2,3	1,2,3,4,5
Потребляемая мощность, Вт	1,35	0,5
Число 8-разрядных РОН	6	8
Число режимов функционирования	1	2

МП предоставляет пользователю возможность выбора выходных управляющих сигналов в соответствии со степенью сложности проектируемой микропроцессорной системы с помощью специального вывода выбора режима конфигурации СО.

В режиме эмуляции МП КР580ВМ80А (устанавливается подачей напряжения 0 В на вывод 11) сохраняются все временные диаграммы и режимы работы МП КР580ВМ80А, а выводы 15 и 28 (10/М и EXM) находятся в третьем состоянии, что позволяет использовать МП КР580ВМ1 вместо МП КР580ВМ80А с минимумом переделок, так как расположение их выводов совпадает.

При подаче напряжения 5 В на вывод 11 МП переводится в другой режим работы. В этом режиме изменяется функциональное назначение ряда выводов МП (указывается в скобках), при этом назначение остальных выводов МП остается прежним. Адресуемая память МП 128 Кбайт.

В МП введены дополнительные регистры общего назначения Н1 и L1, которые можно использовать аналогично регистрам Y и L.

Таблица 3

Основные эксплуатационные параметры

(при температуре от -10 до $+70^{\circ}\text{C}$,
при напряжении питания $+5\text{ В} - \pm 5\%$)

Наименование параметра, единица измерения	Обозначение	Норма		Примечание
		не менее	не более	
Напряжение питания, В	U_{cc}	4,75	5,25	
Входное напряжение, В	U_{ih}	2,0	—	
высокого уровня	U_{il}	—	0,8	
низкого уровня				
Входное напряжение тактовых импульсов, В	U_{cih}	3,9	—	
высокого уровня	U_{cil}	—	0,6	
низкого уровня				
Выходное напряжение, В	U_{oh}	2,4	—	Выходной ток высокого уровня
высокого уровня				$I_{oh} = -400\text{ мкА}$
низкого уровня	U_{ol}	—	0,45	Выходной ток низкого уровня
				$I_{ol} = 2,0\text{ мА}$
				$T = 25^{\circ}\text{C}$
Ток потребления, мА	I_{cc}	—	100	$0,5\text{ В} < U_i < 5,25\text{ В}$
Ток утечки на входах, мА	I_{li}	—		
			$\pm 10,0$	
Выходной ток в состоянии "Выключено", мА	I_{oz}	—	$\pm 10,0$	$0,45\text{ В} < U_o < 5,25\text{ В}$
Входная емкость, пФ	C_i	—	15,0	Измерено относительно корпуса на частоте $f = 1\text{ МГц}$
Емкость входов/выходов, пФ	$C_{i/o}$	—	20,0	
Емкость нагрузки, пФ	C_n	20,0	100,0	

Система команд МП КР580ВМ1 не зависит от режима конфигурации МП.

В XXI ВЕК -
БЕЗ ОРФОГРАФИЧЕСКИХ ОШИБОК!



ORFO IS THE BEST
SPELLCHECKER
FOR RUSSIAN!

**СИСТЕМА ОРФО - ПОСЛЕДНЯЯ РАЗРАБОТКА
ФИРМЫ ИНФОРМАТИК**

ДЛЯ ЧЕГО НУЖНА ОРФО?

ОРФО - система поиска ошибок и опечаток в текстах на русском языке для компьютеров типа IBM PC XT/AT. Кроме

того, система ОРФО эффективно поможет совершенствовать знания в области русского языка. По ряду характеристик ОРФО заметно превосходит аналогичные системы, созданные для других языков. Несмотря на свои уникальные возможности, ОРФО проста в обращении и Вы легко научитесь ею пользоваться.

ОРФО - РЕЗИДЕНТНАЯ ПРОГРАММА, СОВМЕСТИМАЯ С ЛЮБЫМ ТЕКСТОВЫМ РЕДАКТОРОМ.

Пользователь может работать в привычном для него редакторе и при этом использовать все возможности ОРФО.

ОРФО СНАБЖЕНА СЛОВАРЕМ РУССКОГО ЯЗЫКА ОБЪЕМОМ 120000 СЛОВ.

Словарь ОРФО сформирован лингвистами на основе нескольких различных словарей русского языка. В Вашем тексте ОРФО будет “знать” почти все слова (кроме фамилий, редких специальных терминов, нестандартных сокращений, жаргонных слов и т.д.).

ОРФО СПОСОБНА РАСШИРЯТЬ СВОЙ СЛОВАРЬ.

Специальные термины Вы можете сами ввести в словарь ОРФО. Интеллектуальные черты системы проявляются в том, что “обучившись” новому слову, она будет знать все его формы (а их может быть до двадцати). По Вашему заказу можно сформировать специальный профессиональный словарь.

ОРФО НАХОДИТ ОШИБКИ СОГЛАСОВАНИЯ СЛОВ В ПРЕДЛОЖЕНИИ.

Это еще одно следствие интеллекта системы. Если Вы правильно написали слово, но употребили его не в том падеже, роде или числе, то ОРФО может это заметить и указать Вам на необходимость внесения исправлений.

ОРФО УКАЗЫВАЕТ НА ОПЕЧАТКИ В ЗНАКАХ ПРЕПИНАНИЯ И СЛУЧАИ НЕПРАВИЛЬНОГО УПОТРЕБЛЕНИЯ СТРОЧНЫХ И ПРОПИСНЫХ БУКВ.

ОРФО - ЭТО УНИКАЛЬНОЕ СОЧЕТАНИЕ ДОСТИЖЕНИЙ МАТЕМАТИКИ, ВЫСОКОЙ ТЕХНИКИ ПРОГРАММИРОВАНИЯ, ЛИНГВИСТИЧЕСКОГО ИНТЕЛЛЕКТА, СОВРЕМЕННОГО ДИЗАЙНА С МАКСИМАЛЬНЫМИ УДОБСТВАМИ ДЛЯ ПОЛЬЗОВАТЕЛЯ.

СИСТЕМА ПРОВЕРКИ ПЛАТ UTK

Программа UTK - это диспетчер тестов для системы сервисного тестирования плат микропроцессорных систем (МП-систем).

UTK работает на инструментальных машинах типа IBM PC/XT/AT и совместимых с ними (ЕС 1840/41 и т.п.). Инструментальная машина связана с тестируемой МП-системой через один из стандартных портов COM (RS-232).

У тестируемой МП-системы для работы с UTK должны быть:

- ПЗУ не менее 2 Кбайт (для резидентной части UTK);
- ОЗУ не менее 20 байт;
- канал последовательного интерфейса в стандарте Стык 2.

Совместно с программой DBG (системой отладки для МП-систем на основе KM580BM80 (i8080), K1821BM85 (i8085), поставляемой дополнительно) UTK образует комплекс для проверки и определения неисправности в процессорной и в периферийных платах (ОЗУ, ввода/вывода и т.д.) МП-системы. Это позволяет ремонтировать платы и на заводе-изготовителе, и на месте эксплуатации изделия.

Достоинства UTK велики:

- стоит более чем в 10 раз дешевле проверочного стенда;
- тестирует платы МП-системы в реальном времени и к тому же не на проверочном стенде, а непосредственно в устройстве (с реальными шинами, питанием, уровнем помех и т.д.);
- проверка элементов и связей платы - стопроцентна;
- тест проверки платы пишется на языках общего назначения, а не на специализированном языке стенда;
- удаление инструментальной машины от отлаживаемой МП-системы определяется длиной кабеля;
- отыскивает неисправности в МП-платах с запаянным (а не стоящим на колодке) процессором.

Кроме того, UTK:

- загружает с диска предварительно подготовленную технологию проверки платы;
- работает в многооконном режиме;
- автоматически выполняет подтесты в соответствии с технологией;

- автоматически выполняет группу подтестов из всего их массива;

- выполняет одиночный подтест;
- выдает диагностические сообщения при ошибках;
- отслеживает время выполнения подтестов;
- контролирует все операции.

Сопровождение и консультации для работающих с пакетом УТК гарантированы.

По отдельному договору можно разработать тесты для МП-системы заказчика.

Цена системы УТК - 3950 руб.

**Адрес: 127486, Москва, Дегуниная, 1/4,
РАТЕКС/ИНФОАРТ.**

СИСТЕМА ОТЛАДКИ DBG

Система DBG предназначена для отладки в реальном времени микропроцессорных систем (МП-систем) на основе процессоров KM580BM80 (i8080) и K1821BM85 (i8085).

DBG работает на инструментальных машинах типа IBM PC/XT/AT и совместимых с ними (EC1840/41 и т.п.). Инструментальная машина связана с отлаживаемой МП-системой через один стандартный порт COM (RS-232).

У разрабатываемой МП-системы для работы с системой отладки DBG должны быть:

- ПЗУ не менее 1 Кбайт (для резидентной части DBG);
- ОЗУ не менее 20 байт;
- канал последовательного интерфейса в стандарте 'Стык 2'.

Система DBG превосходит традиционные системы отладки (внутрисхемные и шинные эмуляторы).

Преимущества системы отладки DBG:

- более чем в 5 раз дешевле эмулятора;
- работает с двумя типами микропроцессоров (эмулятор с одним);
- удаление инструментальной машины от отлаживаемой МП-системы определяется длиной кабеля (может быть значительным);
- можно ремонтировать оборудование и модифицировать ПО на месте эксплуатации изделия;

- картина работы отлаживаемой МП-системы не искажается;
- отлаживает платы МП-системы с запаянным (а не на колодке) микропроцессором;
- в комплексе с программой УТК (программной системой проверки плат МП системы, поставляемой отдельно) позволяет проверять и ремонтировать платы МП-системы и на заводе-изготовителе, и на месте эксплуатации;
- одна инструментальная машина может отлаживать несколько МП-систем, соединенных между собой по последовательному интерфейсу. Инструментальная машина при этом подключится к любой из них, но работать будет только с одной;
- сервисные возможности системы большие: удобная загрузка файлов, назначение точек останова, пошаговый режим, индикация содержимого памяти и регистров микропроцессора, а также дизассемблированной памяти, модификация памяти и регистров, ассемблерный ввод, многооконный режим индикации, мониторные директивы (заполнить, переместить, сравнить и т.д.), поиск комбинаций байтов, работа с символьными метками загруженного файла, изменение размера и местоположения окон.

Сопровождение и консультации для работающих с системой DBG гарантированы.

Цена системы DBG - 2950 руб.

Справки: 127486, Москва, Дегуниная, 1/4,

РАТЕКС/ИНФОАРТ.

МАШИННАЯ ГРАФИКА НА ЯЗЫКЕ СИ

В 1991 году будут изданы ЧЕТЫРЕ книги (объем каждой книги 12-15 авторских листов) голландского автора Л. Аммерала (перевод с английского В.А. Львова):

1. Принципы программирования в машинной графике.
2. Машинная графика на персональных компьютерах.
3. Интерактивная трехмерная машинная графика.
4. Программирование графики на ТурбоСи.

Описывается комплекс программ машинной графики на языке Си для формирования графической информации и вывода ее на экран монитора персонального компьютера типа IBM PC XT/AT, на матричный принтер или на графопостроитель.

Многочисленные примеры, сопровождаемые полезными советами и полными исходными текстами программ на языке Си, делают эти книги очень полезным учебным пособием для изучения машинной графики и программировании на языке Си.

Ориентировочная цена каждой книги - 15 руб.

Для каждой книги можно заказать комплект программ на дискете за 300...350 руб.

Заказы направляйте по адресу: Москва, 127486, Дегунинская, 1/4, РАТЕКС/ИНФОАРТ.

Образец заявки:

Прошу принять заказ на приобретение книги Л. Аммерала.

Номер тома	Количество экземпляров	Программы на дискете
1.		
2.		
3.		
4.		

Адрес заказчика:

Подпись

О форме оплаты сообщит дополнительное письмо по указанному Вами адресу.

СОВМЕСТНОЕ СОВЕТСКО-АМЕРИКАНСКОЕ ПРЕДПРИЯТИЕ "ICE"

ГАЗЕТА "КОМПЬЮТЕРУОРЛД-СССР"

ПРЕЙСКУРАНТ ЦЕН НА РЕКЛАМНЫЕ УСЛУГИ

	(ч/б)
2стр.(разворот):	30000 руб.
1стр:	15000 руб.
2/3стр:	10340 руб.
1/2стр:	8500 руб.
1/3стр:	6170 руб.
1/4стр:	5000 руб.
1/8стр:	2880 руб.

ГАРАНТИРОВАННОЕ РАЗМЕЩЕНИЕ.

Стоимость по расценкам +10% гарантируют размещение рекламы в специальном месте (за исключением позиций, стоящих выше номинала).

Гарантированное размещение предоставляется с согласия издателя.

СКИДКИ.

Договора, заключаемые в 1991г. предусматривают следующую систему скидок за частоту размещения:

- размещение одного и того же объявления в 2-х или 3-х номерах - 5% (от стоимости размещения 2-х или 3-х объявлений)
- в 4-х или 5-ти номерах - 10%
- в 6-ти номерах и более - 20%

Возможность предоставления иных скидок обсуждается представителем рекламной службы "CW-USSR"

РАЗМЕРЫ СТРАНИЦ ПРЕДОСТАВЛЯЕМЫХ ПОД РЕКЛАМУ:

1стр:	2500х3600
2/3стр:	2500х2400
1/2стр.гориз.:	2500х1800
1/2стр.верт.:	1250х3600
1/3стр.гориз.:	2500х1200
1/3стр.верт.:	833х3600
1/4стр.гориз.:	2500х900
1/4стр:	1250х1800
1/4стр.верт.:	625х3600

НАШ АДРЕС:

129223, г.МОСКВА, ВДНХ, павильон 4 СП"ICE",
фирма "КомпьютерУорлд-СССР"
тел: 181-93-03, факс: 187-88-30.
Расчетный счет: 36200423 во Внешэкономбанке
при ЦОУ г.Москвы, контактор счет: 00165004
МФО 299112



COMPUTERWORLD-USSR

The Publication for the Information Technology Market
RATE CARD

PUBLICATION PROFILE

The IT market in the USSR is characterized by three main factors:
-economic space, highly vast in its absolute dimensions but at the same time practically void of local producers of competitive computers and commercially sold software;

-a sufficiently high level of professional programmers (about 200 thousand) and electronic engineers (about 600 thousand).

"CW-USSR" targeted audience - IMS, DP & General managers of governmental offices of all types (ministries, associations, city councils - 1500-2000), production enterprises (large and medium size - 20000), research institution and universities (2000-2500), Joint Ventures (3000), small and private businesses (20000-25000).

CIRCULATION

CW-USSR has a circulation of 50.000 copies with a readership of over 250.000, 16-24 pages, 12 issues in 1991, then from 1992 - 40 issues.

The publication includes market reviews of the main Soviet regions and worldwide, coverage of technology trends, new applications and industry news, product reviews.

ADVERTISING RATES (black/white)

Double spread:	\$ 7680
Full page:	\$ 3840
2/3 page:	\$ 2798
1/2 page:	\$ 2331
1/3 page:	\$ 1704
1/4 page:	\$ 1393

GUARANTEED POSITION

Earned rate plus 10% guarantees placement of advertisement in a special location (excluding premium positions). Guaranteed positions are subject to the publishers approval.

BLEED

No charge for bleed. Full bleed is available for full pages, spreads and gatefolds.

AD PAGE DIMENSIONS:

Full page:	2500x3600
2/3 page horiz:	2500x2400
1/2 page horiz:	2500x1800
1/2 page vert:	1250x3600
1/3 page horiz:	2500x1200
1/3 page vert:	833x3600
1/4 page horiz:	2500x900
1/4 page:	1250x1800
1/4 page vert:	625x3600



129223 USSR. Moscow,VDNH,pav.4 JV"ICE" firm"ComputerWorld-USSR"
bank account: 57380423/001 in Vnesheconombank USSR
tel: (095) 181-93-03. fax: (095) 187-88-30

НАУЧНО-ТЕХНИЧЕСКИЙ ЦЕНТР

“НОВЫЕ ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ”

ИННОВАЦИОННОГО ОБЪЕДИНЕНИЯ АКАДЕМИИ НАУК СССР

1) разрабатывает информационные и аппаратно-программные комплексы, средства автоматизации, связи и передачи данных для сложных организационно-технических систем на основе локальных сетей, рабочих станций, распределенных систем и сетей ЭВМ, перспективных вычислительных средств и их программного обеспечения с участием совместных предприятий, международных объединений и организаций;

2) создает экспертные системы-оболочки, прикладные экспертные системы и новые информационные технологии на стадиях демонстрационных, опытных, коммерческих и промышленных образцов;

3) разрабатывает естественно-языковые интерфейсы с базами данных, а также экспертные системы баз данных;

4) организует базы данных под конкретные виды управленческой, проектировочной, исследовательской деятельности: охрану окружающей среды, медицину катастроф, экстренные действия в чрезвычайных ситуациях, коммерческие международные системы спутниковой связи и сети средств вычислительной техники;

5) проводит экспертизу информационного и аппаратно-программного обеспечения экспертных систем;

6) оказывает консультационную помощь в разработке информационного и аппаратно-программного обеспечения экспертных систем;

7) производит и реализует товары народного потребления в виде техники, технологии, информации и программной продукции.

Филиалы НТЦ “Инфотех” есть во многих промышленных центрах и регионах СССР.

В НТЦ “Инфотех” работают специалисты высокого класса. Они гарантируют Вам высокое качество продукции, способны разработать проект, провести экспертизу, обучить, помочь консультациями при постановке и разработке проектов. Сопровождение и развитие систем - гарантированы.

НТЦ “Инфотех” предоставляет:

- интегрированную инструментальную среду для проектирования экспертных систем OPS/ST;

- инструментальную экспертную систему диагностики **ДИЭКСПЕРТ**;
- методо-ориентированную экспертную систему для экспертизы и выбора проектов **TET-A-TET**;
- методо-ориентированную систему поддержки процессов принятия решений в экономике и управлении **"Таня"**;
- автоматизированную систему для прогнозирования экономического развития;
- систему-тренажер **"Шехтер"** для обучения иностранным языкам, в частности английскому;
- экспертную систему экспресс-диагностики состояний, требующих неотложной помощи;
- пакет программ для синтеза картографического изображения - **ATLAS (3. 0)**;
- пакет программ для оценки и прогноза радиационной обстановки;
- пакет программ, чтобы оценить, эффективно ли применение новых информационных технологий **AI - EFFECT?**

НТЦ "Инфотех" по заказу разработает аппаратно-программные комплексы по указанным направлениям своей деятельности.

Адрес нашего центрального представительства:
103092, Москва, Б. Сухаревский пер., дом 17, к.17,18;
тел. 208-68-22, 208-69-59.

ИНСТРУМЕНТАЛЬНАЯ ОБЪЕКТНО-ОРИЕНТИРОВАННАЯ ПРОДУКЦИОННАЯ СИСТЕМА OPS/ST

OPS/ST - это интегрированная инструментальная среда для разработки экспертных и других систем, основанных на знаниях. **OPS/ST** поддерживает два способа представления знаний - продукционный и объектно-ориентированный.

OPS/ST в среде языка Смолток:

- четко структуризует предметную область, используя наследование свойств объектов и полиморфизма;
- использует любые объекты языка Смолток в качестве элементов рабочей памяти и расширенный синтаксис условных частей продукций "атрибутными выражениями";
- эффективно реализует операции сопоставления по образцам над Смолток-объектами на основе представления совокупности продукций в виде Rete-сети.

Для OPS/ST характерны гибкий многооконный интерфейс с пользователем на основе меню и развитые средства отладки, включающие в себя трассировку и работу с внутренним представлением продукционных программ.

Система OPS/ST - незаменимый инструмент для построения быстрых прототипов экспертных систем и решения плохоформализуемых задач (диагностики, планирования, принятия решений).

Телефон НТЦ "Инфотех" ИНО АН СССР - 208-69-59, Москва.

АВТОМАТИЗИРОВАННАЯ СИСТЕМА РЕГИОНАЛЬНОГО ЭКОЛОГИЧЕСКОГО ПРОГНОЗА - АСРЭП

АСРЭП предназначена для оценки изменений (за время от 5 до 60 лет) состояния растительности, почв, запасов грунтовых вод, гидросети, загрязненности и засоленности территорий размером от 100 до 10000 км².

АСРЭП применяется для экологической экспертизы крупных хозяйственных проектов, способных повлиять на экологию регионов. При этом можно прогнозировать изменения характеристик лесов, состояния с/х почв и динамики урожая различных культур, объемов водозапасаов (верхних горизонтов, грунтовых вод), расчлененности территории, общих характеристик стока, интенсивности эрозионных процессов, загрязненности (почв, растительности, грунтовых вод) и засоленности (почв и грунтовых вод).

Система учитывает подавляющее большинство видов хозяйственной деятельности: выбросы загрязнителей, лесопосадки, вырубку лесов, внесение удобрений, изменение земельного фонда, изменение пожарной опасности, выпас скота, с/х поливы, водозаборы, дренаж, изменения характеристик гидросети (в результате гидротехнических мероприятий), тенденций изменения климата.

АСРЭП - уникальный инструмент экологической экспертизы. Ни в СССР, ни за рубежом нет даже аналогов. Только благодаря АСРЭП, впервые возможен комплексный экологический прогноз.

АСРЭП реализована на персональных ЭВМ IBM PC/AT. Система, занимая около 320 Кбайт оперативной памяти, рассчитывает один вариант развития региона на срок 30 лет за 50...150 сек.

Телефон НТЦ "Инфотех" ИНО АН СССР - 208-69-59, Москва.

Научно-производственный отдел

“ЭЛЕКТРОНИЗДАТ”

ВНИИ полиграфии

ПРЕДЛАГАЕТ РАЗРАБОТКИ:

- технологию подготовки изданий и изготовление репродуцируемого оригинал-макета на лазерном принтере;
- шрифтовое обеспечение для лазерных принтеров в настольных издательских системах;
- пакет прикладных программ для автоматизации управленческих задач.

ОКАЗЫВАЕТ УСЛУГИ:

- внедрение “под ключ” настольных издательских систем на базе персональных компьютеров;
- передача комплектов технологических инструкций по обработке текста и иллюстраций;
- обучение и консультации по вопросам:
 - обработки текста и иллюстраций,
 - работы на персональных компьютерах, на настольных издательских системах;
- выбор аппаратно-программных средств применительно к конкретным производственным условиям “заказчика”;
- выбор пакетов прикладных программ текстообработки и верстки;
- выполнение производственных заказов (набор).

Оплата услуг производится после окончания работ и их принятию заказчиком по прейскуранту цен, а в режиме “срочно” - по договорным ценам.

Справки по телефону
153-72-17

Адрес: 125130, г. Москва,
Старо-Петровский проезд, д.11.
Проезд: метро “Войковская”



НОВЕЙШИЕ КОМПЬЮТЕРНЫЕ ТЕХНОЛОГИИ ДЛЯ ИЗОБРЕТА- ТЕЛЬСКОЙ ДЕЯТЕЛЬНОСТИ

НОВАТОР - экспертная система для совершенствования известных технических устройств (прототипов) (3,2 тыс. руб.):

- выявляет (в автоматическом режиме) ряд физических противоречий прототипа и определяет способы их устранения;
- содержит базу технических знаний (описание 200 объектов с парными свойствами) и обладает возможностью ее развития.

ПРИНЦИП ДЕЙСТВИЯ - экспертная система для проектирования принципиально новых устройств и технологий (3,5 тыс. руб.):

- содержит более 1500 физических эффектов;
- находит несколько вариантов принципов действия (в виде комбинации физических эффектов) проектируемой технической системы;
- осуществляет поиск физических эффектов по заданным требованиям.

АЛЬТЕРНАТИВА - экспертная система (2,8 тыс. руб.):

- формирует "пространство выбора" - альтернативную древовидную структуру семейства технических систем;
- позволяет делать выбор элементов структуры (в пяти различных режимах) и находить ранее неизвестные их сочетания;
- способствует созданию и развитию зрительных ассоциаций и мышлению образами.

ФСА (функционально-структурный анализ) - пакет программ (3.0 тыс. руб.):

- строит структурную модель исследуемого технического объекта, отражающую взаимосвязь входящих в него элементов;
- формирует функциональную модель взаимосвязи и подчиненности функций элементов объекта;
- анализирует характер и свойства построенных моделей с целью их совершенствования.

Пакеты программ используются на базе персональных компьютеров типа IBM PC XT/AT.

Выше указана стоимость одной защищенной копии. Стоимость каждой последующей копии по 5-ю включительно - 60% от предыдущей.

Демонстрационные пакеты стоимостью от 20 до 55 руб. (в зависимости от количества версий и емкости дискеты) высылаются по получении гарантийного письма или копии платежного поручения. Приобретение документации осуществляется за дополнительную плату.

Предлагаемые технологии несложны в освоении и являются эффективным инструментом повышения конкурентоспособности изделий и увеличения прибыли.

Наш адрес: 119048, Москва, а/я 453

НТК "Метод", тел. 245-46-23

Телекс: 411700 EVRO

факс: 292-65-11 EVRO N 004512

Расчетный счет N 3461299 в Ленинском отделении Мосбизнес-банка г. Москвы, МФО 201188.

СОДЕРЖАНИЕ

Предисловие редактора.....	3
----------------------------	---

СПРАВОЧНИК ПРОГРАММИСТА

Аваков С.М. Настройка операционной среды в системах DOS 3.3 - DOS 4.0.....	5
Ганелин Л.Д. Paradox 3 - лидер, но не в СССР	108
Шерстюк Ф.Н. Вирусы и антивирусы на компьютере IBM PC.....	119
Баевский В.Ю. Конфликт Beta и LANsmart. Как их примирить, не беспокая А.Чижова	138
Некрасов А.А. Подготовка винчестерского диска к работе	144

НАСТОЛЬНЫЕ ИЗДАТЕЛЬСКИЕ СИСТЕМЫ

Гнездилова Г.Г. Национальный алфавит и персональный компьютер.....	156
Глonti Н.Л., Клименко С.В., Малышев В.К., Самарин А.В., Филимонов Б.Б. Метапроект кирилловского алфавита для печатающих устройств с высоким разрешением	186

КРАТКОЕ СООБЩЕНИЕ

Кобылинский А.В., Бестенков В.В., Столяров Е.Я., Темченко В.А. 8-разрядный микропроцессор KP580BM1.....	218
--	-----

Научное издание
Библиотека информационной технологии (БИТ)
Под редакцией Г.Р. Громова
Выпуск 2

Ответственный за выпуск *В.Ю. Кузнецов*
Редакторы *В.М. Ларионова, С.С. Матвеев*
Корректор *Е.М. Кучерявенко*
Технический редактор *Г.А. Полякова*
Обложка художника *А.Я. Коршунова*
ИБ №

Подписано в печать 24.06.91 г. Формат 84x108¹/32. Бумага
офсетная. Гарнитура Таймс. Печать офсетная. Усл. печ. л. 12,6.
Уч. изд. л. 15,2. Тираж 100000 экз (первый завод 40000 экз.)
Заказ № 542 Цена 6 руб.

Если Вы не успели приобрести 1-й выпуск "БИТ" или хотите приобрести дополнительные экземпляры этого выпуска ("БИТ-2"), или хотите оставить заказ на очередной, 3-й выпуск "БИТ", направляйте заявки по адресу:

103031, Москва, ул. Петровка, 15,

магазин N 8 "Техническая книга"

("Книга-почтой")

Цена 6 руб.

COMPUTER КОМПЬЮТЕРУОЛД-СССР **USSR WORLD**

С 1991 г. выходит на русском языке советско-американский еженедельник «КомпьютерУолд - СССР» - газета для деловых людей, программистов и инженеров, чьи профессиональные интересы связаны с информационными технологиями.

«ComputerWorld» - ведущее издание среди двух тысяч газет и журналов, существующих по этой тематике в мире. Газета издаётся в США, Англии, ФРГ, Франции, Японии... - более, чем в двадцати странах. Все национальные редакции «ComputerWorld» объединены в единую информационную систему средствами глобальной электронной почты, что даёт возможность подписчикам «КомпьютерУолд - СССР» оперативно получать материалы из общемировой сети «CW» о ситуации и тенденциях на мировом компьютерном рынке. Значительное место в газете занимают оригинальные статьи советских авторов о становлении этого рынка в СССР.

Стоимость годовой подписки на 1992 год (48 номеров еженедельника):
для организаций - 576 руб.
для частных лиц - 144 руб.

Для тех предпринимателей и специалистов, кто ещё не успел подписаться на нашу газету, сообщаем, что в 1991 г. выйдут 12 номеров (первый номер вышел 18 июля). Для получения полного комплекта из 12 номеров «CW» за 1991 г. необходимо перечислить на наш расчётный счёт: для организаций - 144 руб., для частных лиц - 36 руб.

Расчётный счёт: 36200423 во Внешэкономбанке при ЦОУ Госбанка г.Москвы. Корр. счёт 000165004. МФО 299112.

Заявки на подписку и заверенные копии платежных поручений от организаций или квитанции о почтовом переводе от индивидуальных подписчиков направлять по адресу:

129223, г.Москва, ВДНХ, павильон 4, СП «ICE»,
фирма «КомпьютерУолд - СССР» факс: 187 88 30